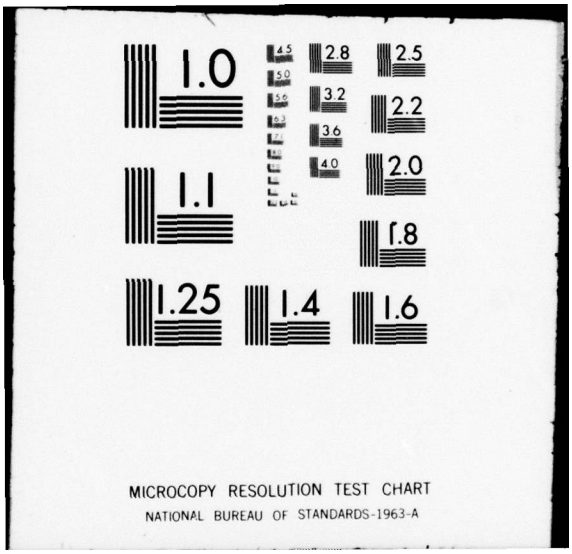
AD-A058 957 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 17/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78

UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-4 NL

1 OF 2
AD
A058957


1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CCTC

DEFENSE COMMUNICATIONS AGENCY

LEVEL III

COMPUTER SYSTEM MANUAL
CSM UM 15-78
VOLUME IV
1 SEPTEMBER 1978

12

COMMAND & CONTROL TECHNICAL CENTER

NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

VOLUME IV
RETRIEVAL AND SORT PROCESSOR (RASP)

USERS MANUAL

(14) CCTC-CSM-UM-15-78-VOL-IV

COMMAND AND CONTROL TECHNICAL CENTER

Computer System Manual Number CSM UM 15-78

(11) 1 Sep[redacted]78

(12) 143p.

(6) NMCS INFORMATION PROCESSING SYSTEM
360 FORMATTED FILE SYSTEM (NIPS 360 FFS).

~~User's Manual~~

Volume IV, Retrieval and Sort Processor (RASP).

(9) user's manual.

AD59 033

SUBMITTED BY:

CRAIG K. HILL
Captain, USA
CCTC Project Officer

APPROVED BY:

FREDERIC A. GRAF, JR.
Captain, U.S. Navy
Deputy Director
NMCS ADP

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.

This document has been approved for public release and sale; its distribution is unlimited.

409 658

ACKNOWLEDGMENT

This manual was prepared under the direction of the Chief for Programming with general support provided by the International Business Machines Corporation under contracts DCA 100-67-C-0062, DCA 100-69-C-0029, DCA 100-70-C-0031, DCA 100-70-C-0080, DCA 100-71-C-0047, and DCA 100-77-C-0065.

CONTENTS

Section Page

ILLUSTRATIONS

Figure Page

TABLES

Table Page

ABSTRACT

This volume defines the capability of the Retrieval and Sort Processor (RASP) component of NIPS 360 FFS. It describes the features, functions, restrictions, language qualifications, and expected output results of this component.

This document supersedes CSM UM 15-74, Volume IV.

CSM UM 15-78, Volume IV - Retrieval and Sort Processor is part of the following additional NIPS 360 FFS documentation:

CSM UM 15-78	Vol I	- Introduction to File Concept
	Vol II	- File Structuring (FS)
	Vol III	- File Maintenance (FM)
	Vol V	- Output Processor (OP)
	Vol VI	- Terminal Processing (TP)
	Vol VII	- Utility Support (UT)
	Vol VIII	- Job Preparation Manual
	Vol IX	- Error Codes
TR 54-78		- Installation of NIPS 360 FFS
CSM GD 15-78		- General Description

RETRIEVAL AND SORT PROCESSOR (RASP)

Section 1

INTRODUCTION

This volume defines the capabilities of the Retrieval and Sort Processor (RASP). RASP retrieves data from files based on English-like retrieval statements. Retrievals can be batched, and data records or periodic sets within data records can be retrieved on information in the fixed and/or periodic fields and/or periodic sets.

The functions of RASP include:

a. Editing and processing of retrieval and control statements;

b. Compilation of retrieval logic to produce machine code for actual retrieval;

c. Determining if an Index Data Set is associated with the data file; and If so, RASP invokes index processing to analyze the retrieval logic and determine if index usage is feasible.

d. Execution of generated code and construction of a Qualifying Data File (QDF) containing records qualified by the retrieval parameters and a Qualifying Record Table (QRT) containing pointers to records in this QDF.

RASP will operate in the batch partition for retrievals run as background processing.

It is possible to specify which periodic sets of a record are to be written on the QDF. A search of a data file can be limited to a specific range of record IDs. Multiple sorts may be specified for each selection, and

different output processing and formatting specifications can be designated for each SORT and SELECT combination.

If either Keyword or Secondary Indexing has been specified for the data file, RASP will analyze the retrieval logic to see if index usage is possible. If it is, the data file search will be restricted to just those records that could possibly qualify instead of the entire data file. If Index Processing is not feasible, RASP will search the entire file, except as restricted by periodic set selection or LIMIT processing.

The Output Processor (OP) and the Quick Inquiry Processor (QUIP) components provide the capability to process the retrieved answer records as specified by the user. Multiple- and single-file retrievals may be accomplished by RASP. However, the merged-file retrieval and output capability must be accomplished by following merged-file retrieval with OP.

The RASP component offers the following capabilities:

a. Batched retrievals against a single file

b. Merged File Retrieval

c. Data record qualification on fixed/periodic/variable information

d. Error diagnostic printout

e. Multiple IF statements within each retrieval

f. Insertion of literals in sort keys

g. Answer record ordering

h. Multiple answer sets from the same retrieval, each with a unique content and sort sequence

i. Storage of retrievals in executable form for later invocation

j. Dynamic analysis of retrieval logic to determine if index processing is applicable. If so, RASP will restrict the detailed logic examination to the potential qualifying records as indicated by Index Processing.

k. Selection of only desired periodic sets with the fixed set rather than whole qualifying logical blocks

l. The ability to use user-written subroutines to qualify data through use of the Function Operator

m. The ability to pass information generated by the user-written subroutines to OP through use of system-provided work areas

n. The ability to change operands at execution time through use of the skeleton query feature

o. The ability to change sort field designations at execution time through use of the skeleton query feature

Retrieval specifications may be from card input or from specifications stored as executable modules on the system or a file library.

RASP will retrieve from data files organized sequentially on magnetic tape or as an indexed sequential data set resident on a Direct Access Storage Device (DASD). RASP will also retrieve data from a Virtual Storage Access Method (VSAM) file.

1.1 Secondary Indexing

Secondary Indexing provides the user the capability to index a data file by the contents of a field other than the Record ID. The primary purpose of the capability is to provide a faster response time for qualifying data records during retrieval. The principles of Secondary Indexing are described in Volume I, Introduction to File Concepts.

The decision whether file indexing information is actually usable is made on the basis of user-supplied retrieval logic at run time. RASP determines if an Index Data Set is associated with the data file based on the existence of Index Descriptor Records in the data file. If there are none, RASP proceeds with its regular functions of retrieving records.

If indexes have been specified for the file, RASP will invoke the Index Processing function to determine if indexing can be utilized or whether the entire data file must be examined. Index Processing decides this based on the logical criteria for data selection as provided by the IF statements. The format of this statement and the criteria used in determining if indexing can be utilized are described in section 2.4.2.2.

Index Processing returns to RASP with an indication of whether the entire data file must be searched, or whether, based on the user logic, only certain records could possibly qualify. The keys of those records are passed to Retrieval Proper and used to access the records and examine them in detail. By restricting the search to just those records that could qualify, RASP can save the time otherwise required to access and examine records that could not possibly qualify.

1.2 Keyword Indexing

The Keyword capability is a text-retrieval capability that provides a method by which records can be accessed and retrieved based upon the known contents of variable length or text data fields. Just as secondary indexing allows access of only those records known to contain the fixed length fields of interest, keyword indexing allows retrieval of records based on the presence of "keywords" within a field. The selection of records based on these "keywords" is accomplished via a KEYWORD statement which is included in the query. It is described in paragraph 2.4.2.2. The principles of keyword indexing are described in Volume I, Introduction to File Concepts.

Prior to employing keyword retrieval, the fields to be examined must have been defined as keyword indexed fields either through File Structuring or the Index Specification utility (see Volume II, File Structuring, Index Definition for Keyword Fields for the methods of defining keyword indexed fields). Depending on when the indexes were specified, either File Maintenance or the utility will maintain the index data set with the fields and their designated keyword values. (See Volume I, Introduction to File Concepts, Keyword Indexing Capability for keyword qualification requirements.) The user determines what values in the field qualify as keywords and from this selection bases the arguments for his retrieval.

Keyword processing is initiated by including a KEYWORD statement in the query. The KEYWORD statement operates on the same level as the secondary LIMIT statement. Only one KEYWORD statement per query is allowed. Structurally and functionally, the statement is similar to the IF statement. Qualification is at the record level; that is, subsets are not flagged although qualification may be the result interrogation of periodic data.

If keyword qualification produces a candidate list of qualifying records but secondary indexing is found unfeasible due to a low percentage of fields in the query being indexed, the keyword list becomes the qualifying list for the query. Otherwise, the list of qualifying records from the KEYWORD statement is merged with the list of qualifying records from the secondary indexed fields in the IF statement to produce one final list containing only those record IDs that contain both qualifying keyword and secondary indexed fields. This list is passed to the retrieval routine to indicate what records are to be accessed and retrieved.

If either source produces no candidates, the retrieval is terminated with an advisory message.

1.3 System Flow

RASP has three sections: the Input Processor section, the Retrieval Proper section, and the Sort section. The flow within each section is discussed briefly in the following paragraphs.

The Input Processor section reads the input stream. It edits each statement and restructures communication records for the rest of the component. It extracts the required information from the File Format Table, resolves all field name addresses and references, and places those data values converted by conversion subroutines in the communication records. It also applies an algorithm to restructure the conditional logic into a bit pattern which can be more readily interrogated using programmed logic. Finally, it converts the retrieval statements into macros which are assembled and link-edited by OS/360 to produce an executable load module. The resulting executable retrieval is stored on the Permanent or Temporary Retrieval Library.

The Retrieval Proper section reads the communication records and calls and executes the code stored on the Permanent or Temporary Retrieval Library. It checks to see if the retrieval, the required subroutines, and the first data set to be retrieved against are available.

If either of the first two are not available, an advisory message is issued, and the job is terminated. If the data set is not available, the operator is notified, and appropriate operator action must be taken.

During initialization, Retrieval Proper determines whether an Index Data Set is associated with the file. If so, it calls the Index Processing function which will determine if indexing information can be utilized. If so, a list of potential qualifiers, called candidates, will be presented to Retrieval Proper which will access only those records.

After all initialization and Index Processing functions are completed, Retrieval Proper is read to pass the retrieval logic against data records. Based on the results

of Index Processing (or absence thereof), Retrieval Proper operates in either candidate-access mode or file-access mode. Candidate-access mode is possible only when an Index Data Set is associated with the data file, when the user has not overridden Index Processing, and when the analysis of user-supplied retrieval logic indicates that Index Processing is feasible. In this mode, RASP accesses only those records indicated by Index Processing as candidates. At all other times, RASP operates in the file-access mode, where all data records are read and examined.

After all initialization has been completed, the data base logical records are read in one at a time. The control field is tested successively against the criteria for the primary and the secondary limits, if any. RASP skips all data records up to the next possible record as determined by primary or secondary limit checking, or the next candidate. Then RASP checks the references to the fixed set, if any. If it fails, all subsequent logical records up to the next fixed set (i.e., the whole logical block) are bypassed. This "early-fail" feature permits more efficient program operation. The same early-fail technique is also applied for each change in periodic set.

The flow sequence continues until every affected set has been processed against the logic of all retrievals. If, after evaluation, the current logic block qualifies against the conditions of a least one retrieval, the logical block is written on the QDF. Entries containing sort keys and QDF pointers are generated as required by the SORT and SELECT statements and written in the QRT and processing resumes as before.

The Sort section invokes the OS/360 Sort/Merge program to sort the records in the QRT.

Section 2

RASP LANGUAGE AND OPERATORS

2.1 RASP Statement Formatting

The language for this component is free-formatted. No constraint is specified for the number of blanks and/or commas separating words in the input stream. Specifications punched in cards may begin in any card column and continue serially through column 71; however, words and literals may not be split between cards. If desired, data elements may be punched one per card, or a number of elements may be punched in a single card. Either format, or any mixture of the two formats, is acceptable. Each language statement must be terminated by a period followed by a blank.

2.2 RASP Definitions

A list of terms and definitions commonly associated with RASP follows:

Action Code -- A field in the QRT sort key used to control the order in which each answer entry record is presented to the Output Processors.

Action Statement -- A direction to perform a specific operation. Action statements in RASP are identified by the following operators:

ADD

DELETE

EXECUTE

SELECT

SORT

Answer Entry Record -- A record in the QRT containing QDF pointers to the beginning of the fixed set and to each periodic set or flagged subset selected for a qualifying logical block. For each retrieval, there will be at least one answer entry record for each sort and select combination within the range of a satisfied conditional statement.

Answer Set -- A group of records in the QRT that are the result of one retrieval and RIT name combination. More than one answer set may be created by one retrieval.

Answer Set ID -- The major control field in a retrieval answer set and consists of the combination formed by the retrieval number and RIT name.

Batch -- A RASP job consisting of two or more retrievals to be executed against the same data base.

Candidate -- A record that may be able to qualify under the retrieval logic, as determined by Index Processing. If indexing can be utilized, according to the logic of the retrieval(s) in the run, the retrieval component will operate in the candidate-access mode; only candidate records falling within the primary and secondary limit ranges will be accessed and examined in detail. Contrast this with file-access mode, which is used when indexing cannot be utilized; every data record will be accessed and examined.

Conditional Statement -- A statement, which taken as a whole, may be true or false according to the rules of

RASP. Conditional statements are identified by the following operators:

FURTHER

IF

LIMIT

KEYWORD

Control Division -- The set of statements which is used to specify the environment in which RASP will operate and to specify action to be taken against the Retrieval Library. The statements in the Control Division are identified by the following operators:

FILE

EXECUTE

LIMIT

DELETE

FILE is required and must be the first statement in any RASP run; the rest are optional. The FILE and LIMIT statements may also appear within the Retrieval Division.

Declarative Statement -- The statements used to identify a retrieval and to specify the files to be searched in single and merged file retrievals. The declarative statements in RASP are identified by the following operators:

TITLE

FILE

File Indexing -- See Secondary Indexing and Keyword Indexing.

Function Operator -- This is a special operator used within the IF or FURTHER portions of a conditional statement which permits the execution of a user-written subroutine as part of the qualification of a data record. The subroutine must always indicate a true or false condition on return and may provide output data in specially provided work areas.

Function Operator Parameter -- This is a data field, literal value or work area which is designated as input (or as output in the case of work areas) to the Function Operator subroutine. The number and order of the parameters is a function of the specific Function Operator subroutine requirements as defined by the user.

Index Processing -- The functions of keyword indexing and secondary indexing that utilizes index information. It determines whether the index information is usable based on the dynamics of the retrieval language at run time.

Index Field -- A field which has been designated as a file index by the Index Specification function.

Indexing -- See Secondary Indexing and Keyword Indexing.

Keyword Indexing - An optional capability whereby a user can index a data file on values (keywords) within a text data field. Variable fields, variable sets and fixed-length alpha fields are subject to this form of indexing. The capability encompases index specification, index maintenance, and index processing. Refer to Volume I, Introduction to File Concepts, for discussion and definition of these terms.

Library Action -- A direction to add, delete, or replace one or more retrievals in the Permanent Retrieval Library.

Logical Block -- A group of records in either the input data base or the QDF, consisting of a fixed set and its associated periodic subsets.

Merged File Retrieval -- A retrieval which involves the search and selection of data records from two or more files. Sort keys are affixed to the QRT entries that describe and point to each data record retrieved. The QRT records are sorted together. Merging pointers to the data have the same effect as if the data had been merged.

Multifile Retrieval -- See merged-file retrieval.

Permanent Retrieval Library -- A disk-resident partitioned data set containing retrievals in executable form. This library may also contain user subroutines, tables, and RITs.

Qualifying Data File (QDF) -- The collection of data records which satisfy the conditions of one or more retrievals built on direct access storage in Record ID sequence.

Qualifying Record Table (QRT) -- The collection of records consisting of sort key, control information, and pointers to each set or flagged subset in the Qualifying Data File which satisfy the conditions of a retrieval. There will be at least one record for each sort and select pair within the range of a satisfied conditional statement for each retrieval in a batch of retrievals. At the completion of the retrievals, the QRT will be sorted on information in the sort key so that it can be processed sequentially by the output processors.

Replaceable Variable -- This term applies to those operands in a conditional clause on a sort statement which have been designated as replaceable in a skeleton query. The operands so designated are replaceable when the retrieval is executed. If not replaced, they default to the originally-defined operand value.

Replacement Variable -- This term applies to those operand values which are defined at execution time to replace those operands which were designated in the skeleton query as replaceable variables.

Retrieval -- The set of declarative, conditional, and action statements furnished by a user which symbolically identifies the retrieval; i.e., assigns it a retrieval ID and labels the answer set; defines the logical conditions for retrieval; and specifies the selection and ordering of the data retrieved. A retrieval may consist of two or more statements containing the following operators.

TITLE

LIMIT

FILE

IF

KEYWORD

FURTHER

SORT

SELECT

The TITLE statement and either the IF or the KEYWORD statement must be present to generate a retrieval. Other statements may be added in selected combinations to specify additional services.

Retrieval Division -- A set of statements which specifies the operations to be performed against the Permanent Retrieval Library (except DELETE) or against one or more data files.

Retrieval ID -- The Retrieval ID consists of the combination formed by the retrieval name and retrieval number as specified on the TITLE statement.

Retrieval Library Action -- The unit of work for RASP which requires action(s) to add, delete, or replace retrievals stored on the Retrieval Library.

Retrieval Name -- A symbolic label assigned by a user which uniquely identifies each retrieval used in the system. The label is specified in the first operation in the TITLE statement.

Retrieval Request -- A statement containing conditions for the qualification of data sets in a retrieval.

RIT -- Report Instruction Table.

RIT Name -- The label which uniquely identifies a group of specifications (RIT) that, when executed, produces a required output format.

Secondary Indexing -- An optional capability whereby a user can index a data file based on the contents of a field other than the Record ID. The capability encompasses Index Specification, Index Maintenance, and Index Processing. Refer to Volume I, Introduction to File Concepts, for discussion and definition of these terms.

Single-File Retrieval -- One or more retrievals to be executed against one file in a particular job.

Skeleton Query -- This is a query which contains operands in the condition portion of the query and/or in the sort statement(s) which have been designated (by an underscore character, Ø-5-8 punch, prefixed to the operand) as being replaceable. The skeleton query is compiled and stored on the Permanent Retrieval Library. It can be executed, replacing any or all of the operands which were designated as replaceable. Any operand which is not replaced will default to its original operand value.

Work Area -- There are five areas available for use; each area is four bytes in length and may be referenced, using system names WORK1 through WORK5. The work area can be designated as a parameter to the Function Operator, as a value in a condition clause or as a field in a SORT statement. Data is placed in the work area through use of the Function Operator and when

referenced, is automatically transferred to the QRT, thus making it available to the Output Processor for display or further processing.

2.3 Control Division

Statements used as input to RASP have been classed into two divisions: Control Division and Retrieval Division. The Control Division consists of those RASP statements defining the RASP job environment and listing the delete actions to be taken against the Retrieval Library. The Control Division statements are:

FILE

LIMIT

EXECUTE

NOTE

DELETE

2.3.1 Run Initializing Operators

The Control Division FILE statement is always required to initialize a RASP run; several other run-initializing operators may be added if desired. The LIMIT statement may be added to restrict the search of the data file to those logical blocks in a specific range of record IDs. The EXECUTE statement may be used to initiate a Retrieval Library search for one or more retrievals and cause them to be loaded and executed. The NOTE statement allows the user to comment his source listing. The DELETE statement allows the user to delete those retrievals previously placed on the Permanent Retrieval Library.

2.3.1.1 FILE

The FILE statement must be the first Control Division statement. The FILE statement designates the file to be

employed in a single-file retrieval and a file in a merged-file retrieval. The file name specified in the Control Division FILE statement operand may be referenced explicitly in the retrieval by use of a Retrieval Division FILE statement. The file name specified on the FILE statement must correspond to the unqualified data set name or the last segment of a qualified data set name designated on the DD statement.

The format is:

FILE filename.

A single-file job is a job in which all FILE statements name the same file. Within the single-file job, a FILE statement may follow each TITLE statement (optionally).

A merged-file job is a job in which more than one file is named within a single retrieval. A maximum of 10 FILE statements are permitted in a merged-file retrieval. One of the files must be named in the FILE statement in the Control Division and that must be the first Control Division statement.

Example

FILE TEST36Ø.

Comment: This statement specifies that the statements which follow pertain to the file named TEST36Ø.

There may be a mixture of tape and disk files in one job. A single QDF-QRT is produced for a merged-file job, but the QRT is sorted so that outputs will be separated properly.

Note: In a merged-file job there is only one retrieval; i.e., only one TITLE statement.

2.3.1.2 LIMIT

The LIMIT statement allows the user to restrict the search of a data file to logical blocks within a specified range of record IDs. The condition of a LIMIT statement is applied in one of these modes called START, STOP, or SEARCH. This capability is effected by screening the high-order positions of the record ID for the data field or group value(s) specified in the LIMIT statement, and using the resulting condition to:

a. Initiate a detailed logical search of subsequent records (START mode)

b. Terminate the search against the file (STOP mode)

c. Perform detailed search within the range established by the first and second occurrences of the record ID field value specified in the LIMIT statement (SEARCH mode). This search includes the values specified as a part of the detailed search.

The format is:

```
                 ┌field-name┐
LIMIT    [IF]    │          │    [partial-field]     [subroutine-name]
                 └group-name┘
     ┌                         ┌                   ┐                               ┐
     │                         │     literal       │                               │
     │  GT                     │                   │  [STOP].                      │
     │                         │self-defining term │                               │
     │                         └                   ┘                               │
     │                         ┌                   ┐                               │
     │                         │     literal       │                               │
     │  GE                     │                   │  [START].                     │
     │                         │self-defining term │                               │
     │                         └                   ┘                               │
     │              ┌            ┐┌   ┐┌            ┐                               │
     │              │     literal││AND││   literal  │                              │
     │  BT          │            ││or ││            │ [SEARCH].                    │
     │              │self-defining││ / ││self-defining│                             │
     │              │    term    ││   ││    term    │                              │
     └              └            ┘└   ┘└            ┘                              ┘
```

where

GT is stop-when-greater-than,
GE is start-when-equal-to-or-greater-than,
BT is search-for-equal-to-or-between.
STOP, START, and SEARCH are optional words which may be used for clarity, but may not be used to change the meaning of an operator.
The . is the statement terminator.

The placement of the LIMIT statement determines the effect on the retrievals in the job. If the statement appears in the Control Division, it is a primary LIMIT statement and applies to all retrievals in a job, including those specified in the EXECUTE statement. A primary LIMIT statement may be placed anywhere in the Control Division; however, only one primary LIMIT statement may appear in a single file job. Moreover, the primary LIMIT statement is not permitted in merged file jobs. Finally, primary LIMIT statements are compiled and executed but cannot be placed on the Retrieval Library.

If the LIMIT statement appears in the Retrieval Division, it is a secondary LIMIT statement and applies only to a specific retrieval.

The secondary LIMIT statement follows the TITLE statement if there is no FILE statement; otherwise it follows the FILE statement.

For example:

```
TITLE     or   TITLE     or   TITLE
LIMIT          FILE           FILE
               LIMIT          LIMIT
                                .
                                .
                                .
                              FILE
                              LIMIT
```

If the secondary LIMIT statements are used without a primary LIMIT statement, secondary conditions will be imposed upon each data record affected by that retrieval.

If a secondary LIMIT statement is used in addition to the primary LIMIT statement, the restrictions imposed by the secondary LIMIT statement will be imposed only if the current record falls within the conditions imposed by the primary LIMIT statement.

Secondary LIMIT statements will be compiled and placed on the Permanent Retrieval Library with the rest of the retrieval logic. It will be executed each time the retrieval is executed.

To remove the secondary LIMIT function from a retrieval on the Permanent Retrieval Library, the retrieval must be recompiled. The LIMIT statement must be omitted, and a new TITLE statement specifying REPLACE (retrieval name) as an operand must be included.

The maximum number of LIMIT statements permitted in a single file retrieval job is 96 (one per retrieval), while the maximum number of secondary LIMIT statements permitted in a merged file retrieval is 10.

Only field values contained in the Record ID may be addressed via the LIMIT statement. Also, only the high-order field (or fields, if a Record ID control group is named) may be addressed by the LIMIT statement. For this reason, when a partial-field option is used, the first value must be a 1. Any other value will produce an error message.

Example 1

```
LIMIT IF UIC GT J00026.
```

Comment: A STOP condition is implied by the greater-than relational operator.

Detailed conditional logic will be applied to each record from the beginning of the file and will stop with a value greater than J00026.

The effect is the same as reaching an END-OF-FILE condition.

Example 2

```
LIMIT IF UIC GE M00019 START.
```

Comment: A START condition is imposed by the 'greater than' or 'equal to' relational operator. The search will begin with the record which contains a value of M00019 or greater in the Unit Identification Code group.

Example 3

```
LIMIT IF UIC BT M00007/W00003 SEARCH.
```

Comment: A BETWEEN condition is imposed.

All records whose control group contains values in the range M00007 and W00003 inclusive will be retrieved using the IF/FURTHER conditional logic.

Example 4

```
FILE TEST36Ø.
LIMIT IF UIC BT K$ AND V$.
TITLE SORV26/Ø312.
LIMIT SERV BT J AND M.
IF UNLVL EQ SQ AND MEDEP EQ Ø
AND MEQPT NE Ø AND MECI EQ A.
SORT ATTACH, SERV, UNLVL.
TITLE SORV27/Ø313.
IF UNLVI EQ SQ AND MECL 1/1 EQ G M.
SORT ATTACH, SERV, UNLVL.
```

Comment: This batch contains both a primary LIMIT and a secondary LIMIT. The primary LIMIT affects both retrievals, restricting file reading to those records in which the first character of the Record ID is K, L, M, or V. (The '$' symbol indicates the end of a partial field.) In addition, the first retrieval is restricted to a test of those records for which the major field in the control group (high-order position of Record ID) contains a value between J and M inclusive; thus the combination of primary and secondary LIMITS restricts the first retrieval to those logical blocks whose Record ID begins with K through M.

Example 5

```
LIMIT IF SERV BT M/M.
```

Comment: The between relational operator may be used to restrict application of conditional logic to a specific value. In this example only those records containing a "M" in the high-order position of the record identification field will be processed.

2.3.1.3 LIMIT Processing and Index Processing

Index Processing will recognize any primary LIMIT specified and will consequently eliminate candidate records not falling within the primary range. Records within this range may be within the secondary LIMIT of some retrievals, but not within that of others. Index Processing will not be concerned with any secondary LIMIT, but will present all candidates within the primary range for examination. Those records outside the secondary range for each retrieval will then be eliminated by normal LIMIT processing.

Assume in the following example that the search will be restricted to reports in which the field CNTRY contains the value "US".

```
FILE TEST360.

LIMIT IF UIC BT K$ AND V$.

TITLE SORV28/0314.

LIMIT SERV BT J AND M.

IF CNTRY EQ US.

SORT ATTACH SERV UNLVL.

TITLE SORV29/0315.

IF CNTRY EQ US AND MECL 1/1 EQ G,M.

SORT ATTACH SERV UNLVL.
```

Comment: This retrieval contains both a primary LIMIT and a secondary LIMIT. The primary LIMIT affects both retrievals, restricting file access to those records in which the first character of the Record ID is in the range K to V. In addition, the first retrieval is further restricted to examining only those records whose service field (SERV) is in the range J to M. The primary and secondary LIMITs combine to restrict the first retrieval to those data records whose Record ID begins with K through M.

Index Processing will recognize only the primary LIMIT. Only those candidates whose beginning Record ID character is in the range K through V will be presented. For the first retrieval, those records outside the range of J through M (the range of the secondary LIMIT) would be eliminated by ordinary LIMIT processing.

2.3.1.4 EXECUTE

The EXECUTE statement allows the user to specify the processing of retrieval(s) stored on the Permanent Retrieval Library.

The format is:

```
EXECUTE retrieval name (replacement variables),
        retrieval name (replacement variables)...
```

This statement is required only if one or more retrievals are to be executed from the Permanent Retrieval Library. Each library retrieval must have been compiled for one of the files specified in the present retrieval. The statement must be terminated by a period.

Example 1

```
EXECUTE TSTFØ2R.
```

Comment: Illustrates execution of a single retrieval.

Example 2

```
EXECUTE TSTFØ5R (ARMY=NAVY),TSTFØ7R (ARMY=USAF),
        TSTFØ9R (PLANE=F111).
```

Comment: Illustrates execution of multiple retrievals with definition of replacement variables for a skeleton query.

The EXECUTE statement is used to define replacement variables when executing a skeleton query. Replacement variables are specified following the retrieval name. The

replacements are enclosed in parentheses, and the EXECUTE statement is terminated with a period. The original operand, as designated in the stored skeleton query, is used as a keyword. This is followed by an equals character (=) and the replacement operand. There are no spaces between the keyword, the equals character, and the replacement operand. When an operand is being replaced by multiple operands, the replacement would appear as keyword, equals character, open parenthesis, operands (separated by commas or blanks), closed parenthesis. All replaceable variables in the skeleton do not have to be replaced. Those which are not replaced will default to the originally-specified value in the stored query. (See section 2.6 for instructions on defining a skeleton query.)

The keyword operands must be coded exactly as they appear in the stored skeleton query. The replacement variables have a form similar to the original operands. For example, if a field name prefixed with an ampersand is being replaced, then the replacement variable (field name) must be prefixed with an ampersand. Operands with embedded blanks must be enclosed in quotes. Conversion subroutines, tables or partial field notation cannot be specified on the EXECUTE statement.

Assume for the following EXECUTE examples that the sample skeleton query shown below was previously compiled and stored on the Permanent Retrieval Library. Note particularly those items which are preceded by the underscore character, and their relation to the examples.

```
FILE TESTER.
TITLE RET1/01 NO-GO.
IF MEQPT EQ _TANK AND MEADA LT _&MEPSD.
SORT SERV.
```

Example 3

```
EXECUTE RET1(TANK=GUN,&MEPSD=&MERDY).
```

Comment: Where the original value TANK appeared, preceded by an underscore, TANK will be replaced by GUN, and &MEPSD, preceded by an underscore, will be replaced by &MERDY.

A multiple value replacement must be enclosed in parentheses.

Example 4

```
EXECUTE RET1(TANK=(GUN,SHIP,PLANE)).
```

Comment: Where the original value TANK appeared in the query, it will be replaced by the multiple value of GUN, SHIP, and PLANE; &MEPSD would not be replaced.

A skeleton may be executed more than once with different replacement values within the same execution for single-file retrievals (the multiple execution of merged-file retrievals is equivalent to batching merged-file retrievals, which is illegal). However, the retrieval number for the second and succeeding executions of that skeleton will automatically be suffixed with an alpha character beginning with the character 'A' and continuing in sequence.

Example 5

```
EXECUTE RET1 (TANK=GUN,&MEPSD=&MERDY),
RET1(TANK=SHIP,&MEPSD=&MEREQ).
```

Comment: The answer IDs to publish the answer sets resulting from the preceding example would be 'Ø1' for the first answer set and 'Ø1A' for the second.

A value designated as replaceable can be omitted by designating a dollar sign mask with no other designation.

Example 6

```
EXECUTE RET(TANK=$,&MEPSD=&MERDY).
```

Comment: Where TANK was designated as replaceable, the condition will be bypassed. The conditional statement to be executed will therefore be:

```
IF MEADA LT &MERDY.
```

If the omitted value is within a Function Operator parameter list, the length for that field will be set to zero. The user must recognize this within his subroutine (see section 2.7).

2.3.1.5 NOTE

The NOTE statement allows the user to comment his listing of source statements. Every character between NOTE and the terminator (period blank) is considered part of the comment. Since NOTE statements play no part in RASP logic, they may appear anywhere in the RASP input.

2.3.2 Library Action Operators

The library action operators provide the facilities to add, replace, and delete retrievals from the Permanent Retrieval Library. Two of the operators, ADD and REPLACE, are used as operands in the TITLE statement since a complete retrieval must be supplied to effect either of these operations. A complete discussion of these two operators is given in subsection 2.4.1.1.

2.3.2.1 DELETE Statement

The DELETE statement designates a retrieval whose name and associated library information is to be deleted from the RASP Permanent Retrieval Library.

The format is:

DELETE retrieval-name,..., retrieval-name.

Any number of DELETE statements may be used, but no more than 256 retrieval names may be deleted in one RASP execution.

Example 1

```
DELETE TSTFØ1.
```

Comment: Illustrates one retrieval deleted.

Example 2

```
DELETE TSTFØ1, TSTFØ1, TSTFØ2, TSTFØ9.
```

Comment: Illustrates more than one retrieval deleted.

2.4 Retrieval Division Statements

The Retrieval Division consists of those statements used to identify each retrieval, specifies the conditions for data selection, and defines the parts of the record to be included in the answer set and the ordering or arrangement to be applied to the data retrieved. Of the following Retrieval Division statements, only the TITLE and either the KEYWORD or the IF statements are required for a complete retrieval specification.

The Retrieval Division statements are:

```
TITLE
FILE
KEYWORD
NOTE
LIMIT
IF
FURTHER
SORT
SELECT
```

The statements will be discussed in the following sections in the order cited. The NOTE statement may occur at any point; however, the arrangement of other statements must conform to certain allowed sequences. Table 1 summarizes the correct sequence for retrieval statements.

2.4.1 Retrieval Initializing Operators

The Retrieval Initializing Operators, within a retrieval, are TITLE, FILE, and NOTE.

2.4.1.1 TITLE

The TITLE statement allows the user to label the retrieval and assign a label to the answer set; to test the retrieval for compilation errors or default to the assemble-and-go mode; to add the retrieval to the Permanent Retrieval Library or to replace existing retrieval with this one: and to specify an alternate method of scanning subsets.

Table 1

SEQUENCE MATRIX FOR RETRIEVAL STATEMENTS

	Possible Next Statement							
Current Statement	TITLE	FILE	LIMIT	KEYWORD	IF	FURTHER	SORT	SELECT
TITLE		X	X	X	X			
FILE			X	X	X			
LIMIT				X	X			
KEYWORD	X	X			X			
IF	X	X			X	X	X	X
SORT	X	X			X	X	X	X
SELECT	X	X			X	X	X	
FURTHER	X	X			X		X	X

X = Correct Sequence

RETRIEVAL AND SORT PROCESSOR (RASP)

The format is:

TITLE retrieval name/retrieval number [RIT-name] [NOGO SUBSCAN]

[ADD,]
[REPLACE] [retrieval name.]

where

TITLE (required) -- Is the retrieval initializing operator and must be the first word in the statement.

Retrieval name (required) -- Is the retrieval name consisting of one to seven alphameric characters, the first of which must be alphabetic.

/Retrieval number (required) -- Provides the 1- to 4-digit retrieval number used to label the answer set produced for the output processors.

RIT-name (optional) -- Is the name of the RIT that will be used to process this retrieval.

NOGO, NO-GO, NO GO (optional) -- Specifies that the retrieval is to be compiled but will not be executed.

SUBSCAN (optional) -- Specifies that all qualifying subsets should be flagged.

ADD, REPLACE -- Is included in the operand field, according to the function to be performed:

ADD: The ADD option indicates that the retrieval is to be added to the Permanent Retrieval Library.

REPLACE: The REPLACE option indicates that the retrieval is being entered in its entirety as a replacement for an old retrieval. The retrieval ID specified must already exist in the library.

Note: Only one function (ADD or REPLACE) can be used at a time.

.(required) -- Is the statement terminator.

The TITLE statement is always the first statement of a retrieval. The retrieval ID, consisting of retrieval name/retrieval number, must be the second operand of the TITLE statement. Any operands that follow may appear in any order.

The TITLE statement occurs once for each retrieval. The retrieval name and RIT name must be unique within the system; the retrieval number must be unique within the run.

Example 1

```
TITLE TSTF/101.
```

Comment: Labels the retrieval as TSTF and the answer set as Ø10Ø. Illustrates assemble-and-go with no library actions and no RIT specified.

Example 2

```
TITLE TSTFER/23 DIAGNOS.
```

Comment: Labels the retrieval as TSTFER and the answer set as 0023DIAGNOS. It accepts the assemble-and-go mode by default and defines that the answer set labeled ØØ23DIAGNOS will be processed by the RIT named DIAGNOS.

Example 3

```
TITLE TESTFEST/11 RPTSOFA NO-GO.
```

Comment: Labels the retrieval as TSTFEST and the answer set as ØØ11RPTSOFA. NO-GO specifies that this retrieval will be compiled but will not be executed.

Example 4

```
TITLE TSTTEST/22 REPLACE TSTFEBS DIAGNOS.
```

Comment: Labels the retrieval as TSTTEST. Labels the answer set as ØØ22DIAGNOS. It requires a library action to replace an existing retrieval having the name TSTFEBS with a new version and to process the results using a RIT name DIAGNOS. The assemble-and-go mode will be taken by default. Note that if the retrieval to be replaced (TSTFEBS) is not found on the Retrieval Library, RASP will still add the TSTTEST retrieval to the library.

Example 5

```
TITLE TSTFST/22 REPLACE TSTFEBS DIAGNOS.
```

Comment: Same comments as above except that a different retrieval is replacing TSTFEBS.

Example 6

```
TITLE TSTFRAA/1Ø1,TSTFRUN, ADD.
```

Comment: Labels retrieval as TSTFRAA, the answer set as Ø1Ø1TSTFRUN; specifies that the RIT to be used is TSTFRUN and this retrieval is to be added to the Retrieval Library. The default option of assemble-and-go is desired. In this case the RIT name precedes the library operator ADD; however the sequence of these two operands could have been reversed. Note that if a retrieval named TSTFRAA already exists on the Retrieval Library, this retrieval will not be added.

Example 7

```
TITLE TSTFRE/1Ø1,REPLACE TSTFER, NO-GO DIAGNOS.
```

Comment: Labels retrieval as TSTFRE and answer set as Ø1Ø1DIAGNOS and specifies that TSTFRE is to replace TSTFER. The mode is compile only and the RIT name DIAGNOS will be used to process the answer set produced by this retrieval. If the test is successful, the user only need pull the NO-GO operand. Note that DIAGNOS appears last and is followed by a period which terminates the statement.

2.4.1.2 FILE

The FILE may be repeated in a single file retrieval in which case it must follow the TITLE statement and must name the same file specified in the Control Division.

The FILE statement is used in a merged file retrieval to indicate that the conditional logic immediately following applies to the file named in the operand of this statement and overrides the primary FILE named in the Control Division FILE statement. If the secondary FILE statement is used in a retrieval, there must be only one retrieval in the job; i.e., merged file retrieval, and there may be up to 10 FILE statements.

The format of the secondary FILE statement is:

```
FILE      filename      .
```

where

FILE (required) - is the statement identifier.

filename (required) - is the 7-character file name which corresponds to the unqualified data set name or the last segment of a qualified data set name to be processed by this retrieval.

. (required) - is the statement terminator.

2.4.1.3 NOTE

The NOTE operator is discussed in subsection 2.3.1.4.

2.4.2 Conditional Statements

The conditional statements within a retrieval are LIMIT, KEYWORD, IF, and FURTHER.

2.4.2.1 LIMIT

The LIMIT statement is described in subsection 2.3.1.2.

2.4.2.2 KEYWORD

To employ keyword retrieval, a KEYWORD statement must be included to provide the logical criteria for data selection. The KEYWORD statement consists of conditional clauses, logical connectors and parentheses. Only one KEYWORD statement is allowed for a query. It may be the only conditional statement in a query.

The format it:

KEYWORD clause [AND / OR] clause... [AND / OR] clause.

where

KEYWORD(required) - is the statement identifier and must be the first word in the statement.

Clause(at least one required) - is a true assertion. The method of coding clauses is described below.

AND, OR(optional) - defines the relation between clauses (or groups of clauses) when the KEYWORD statement contains more than a single clause.

.(required) - is the statement terminator.

2.4.2.2.1 Condition Clauses and Keyword Indexing

Coding of a keyword condition clause requires use of a keyword indexed field, the INCLUDES relational operator, and a value or multiple values. The complete format of one clause is:

fieldname / groupname INCLUDES data value(s)

where

fieldname,groupname - identifies the name of a fixed-length alpha field, variable-length field, or variable set that has been defined as a keyword indexed field and is to be tested against the data value(s).

INCLUDES - is the relational operator stating that the data value must be included within the specified field.

data value - is either a literal or self-defining term which is a keyword for the field. There may be multiple data values.

2.4.2.2.2 LIMIT Processing and Keyword Indexing

The KEYWORD statement functions on the same level as the secondary LIMIT statement. The list of candidates from keyword index processing is merged with the list from secondary index processing. If there is no secondary indexing, the keyword list becomes the candidate list for retrieval. Index processing is not concerned with a secondary LIMIT statement. All candidates within the primary range are presented for retrieval examination. Those records falling outside the secondary range are eliminated by normal LIMIT processing. Index processing does, however, recognize any primary LIMIT specified and eliminates candidate records that do not fall within the primary range.

```
FILE TEST360.
TITLE Q1/1.
LIMIT IF SERV BT J/M.
KEYWORD COMMENT INCLUDES AIRCRAFT.
IF UNFLG EQ K.
```

UNFLG is not an indexed field, therefore, the candidate list presented for retrieval will consist of all the record IDs that have a COMMENT field which includes AIRCRAFT. LIMIT processing will eliminate all records that have a SERV field not between J/M. Qualifying records will be those

that have a SERV between J and M, and COMMENT field containing AIRCRAFT and UNFLG field equal to K.

```
FIELD TEST360.
LIMIT IF UIC BT K$ AND V$.
TITLE Q2/2.
LIMIT IF SERV BT J/M.
KEYWORD COMMENT INCLUDES AIRCRAFT.
IF UNFLG EQ K.
```

Index processing will recognize the primary LIMIT statement so that the only records that will become candidates for retrieval will be those which have a letter between K and V as the first character of their record ID and the COMMENT field containing AIRCRAFT. The final qualification will be to follow the same criteria as the previous example.

2.4.2.2.3 Examples Using Keyword Indexing

The KEYWORD statement invokes index processing to break down the statement into clauses and to match the data value in the clause against the keywords for that field in the index data set. The keywords consist of previously selected value(s) from the data field that are likely subject candidates for retrieval. Once a keyword field qualifies a record for retrieval, no further examination of that field is necessary.

Example 1

```
KEYWORD REMARKS INCLUDES NMCSSC.
```

In this example, REMARKS is a variable field in the data file that has also been defined as a keyword index. Only those records that have a REMARKS field containing the keyword NMCSSC will be included in the candidate list. In this case, since there was only one value specified, all candidate records will qualify for retrieval.

Example 2

```
KEYWORD REMARKS INCLUDES NMCSSC, AOC OR VSET1 INCLUDES
'NOVEMBER, 1972'.
```

In this example, REMARKS is a variable field, VSET1 is a variable set and both have been defined as keyword indexes. All records that have a REMARKS field containing either NMCSSC or AOC or a VSET1 containing NOVEMBER, 1972 will be presented as candiates. This literal must be bounded by quotes in order to have the special character of a comma and the blank included as part of the value search.

Example 3

```
KEYWORD (REMARKS INCLUDES VIETNAM AND TYPE INCLUDES F-111) OR
(COMMENT INCLUDES AIRCRAFT AND HOME INCLUDES N).
IF DATE EQ JUL72.
```

In this example, REMARKS and COMMENT are variable fields, TYPE and HOME are fixed-length alpha fields, and all have been defined as keyword indexes. DATE is a fixed-length field defined as a secondary index. The following records would become candidates for retrieval: those having REMARKS containing VIETNAM, TYPE containing F-111, COMMENT containing AIRCRACT, HOME equal N and DATE equal to JUL72.

Example 4

```
KEYWORD REMARKS INCLUDES TECHNICIAN.
IF RANK EQ 'E-7'.
```

In this example, RANK is not an indexed field. The candidate list would consist of those records containing a REMARKS field which includes the value TECHNICIAN. These records will be accessed to examine the RANK field. The qualifying retrieval records will be only those that have both REMARKS containing TECHNICIAN and RANK equal E-7.

Example 5

```
FILE TEST360.
TITLE Q3/3.
```

```
KEYWORD COMMENT INCLUDES PERSONNEL.
IF CNTRY EQ UK.
TITLE Q4/4.
KEYWORD COMMENT INCLUDES BASE.
IF CNTRY EQ 'GE'.
```

One KEYWORD statement per query is allowed. COMMENT is a keyword indexed field, CNTRY is a secondary indexed field. The first query will produce output of all records that have COMMENT field containing PERSONNEL and CNTRY field equal to UK. The second query will produce output of all records that have COMMENT field containing BASE and CNTRY field equal to GE.

2.4.2.3 IF

Every retrieval which does not have a KEYWORD statement must have an IF statement. This statement provides the logical criteria of data selection for fields that are not keyword indexes. The IF statement consists of conditional clauses, logical connectors, and parentheses.

The format is:

IF clause [AND / OR] clause... [AND / OR] clause.

where

IF (Required) -- Is the statement identifier. It must be the first word in the statement.

AND, OR (Optional) -- Defines the relation of two clauses when the IF statement contains more than a single clause.

Clause (Required) -- Is a true or false assertion, such as: The boy's name is John; the boy lives in Missouri; the boy is under 21. The method of coding clauses for RASP to evaluate is described below. Clauses may be enclosed in parentheses to define complex relations:

IF (clause-1 OR clause-2) AND (clause-3 OR clause-4). is equivalent to

IF clause-1 AND clause-3 OR clause-1 AND clause-4 OR clause-2 AND clause-3 OR clause-2 AND clause-4.

. (Required) -- Is the statement terminator.

There are three types of conditional clauses. The standard type of clause consists of a file field, a relational operator, and a value(s). The second uses the ABSENT operator and consists solely of a file field and the operator ABSENT. The third is referred to as a Function Operator and causes the execution of a user-written subroutine as part of the qualification of the data record.

The first and third types of clause may reference an index field, and therefore cause indexing information to be used. How Index Processing interprets the retrieval logic and the rules by which it determines if indexing information can be utilized are discussed in section 2.4.2.3.2, Condition Clauses and Secondary Indexing. In sections 2.4.2.3.1, Standard Condition Clause, and 2.4.2.3.4, FUNCTION Operator, only specific points covering Secondary Indexing, such as restrictions, will be discussed.

2.4.2.3.1 Standard Condition Clause

Coding of a standard condition clause requires use of a file field, a relational operator and a value or multiple values such as:

PERS EQ 16

This asserts that the specified file field contains a data value of 16, which may be true or false. RASP generates coding to determine the truth or falsity of each clause and the combined truth or falsity of the IF statement. The complete format of one clause is:

```
       ┌field-name┐
[ANY]  │          │[partial-field] [subroutine-name] [NOT]
       └group-name┘

                              ┌ data values        ┐
       relational operator    │ self-defining term │
                              └ literal            ┘
```

where

ANY (optional) -- Is used to modify the conditional logic so that successive terms against the same periodic set are not evaluated within one subset at a time. (Ordinarily, when two or more conditional clauses make reference to the same periodic set, all clauses are evaluated within one subset at a time. The keyword ANY modifies conditional logic to allow the conditions to be met within the entire set rather than one subset.) For example, consider the TEST36Ø file. To test for a unit with <u>both</u> B-52 and C-5A aircraft:

IF MEQPT EQ B-52 AND ANY MEQPT EQ C-5A.

Note: Without the ANY modifier in the above example, both conditions would have to be met within the same subset, which is not possible. Use of the SUBSCAN option on the TITLE statement modifies subset scanning such that the "OR ANY" condition is treated as an "OR" condition; i.e., successive terms against the same periodic set are evaluated one subset at a time.

Field-name, group-name (required) -- Identifies the character string in the record to be tested against the data value. When used with either the circle or overlapping polygon relational operator, the field-name or group-name must reference an internal coordinate mode character string.

Partial-field notation (optional) -- Specifies the portion of the file field or group character string to

be used in the compare. Partial-field notation cannot be used with the circle or overlapping polygon relational operators. Partial-field notation is permitted into but not out of a subroutine or table. Partial-field notation may not be specified on binary, or coordinate fields. Partial field notation, if used, will nullify Index Processing for the clause.

Subroutine-name (optional) -- Is the name of a subroutine used to convert the data value (s) following the relational operator to internal format. Subroutine conversion must not be specified with the circle and overlapping polygon relational operators. It is also not allowed with CONTAINS relational operator.

NOT (optional) -- Reverses the truth value of the relational operator.

Relational operator (one required).

LT, LESS, BEFORE, EARLIER -- Are relational operators signifying 'less than'.

LTE, LE -- Are relational operators signifying 'less than or equal to'.

GT, GREATER, AFTER, LATTER -- Are relational operators signifying 'greater than'.

GTE, GE -- Are relational operators signifying 'greater than or equal to'.

EQ, EQUAL, EQUALS, EQUALING -- Are relational operators signifying 'equal to'.

NE -- Is a relational operator signifying 'not equal to'.

BT, BETWEEN -- Are relational operators signifying 'between (and including)'.

CONTAINS -- Is a relational operator which is used to scan the referenced field or set for the designated

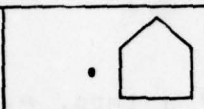
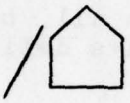
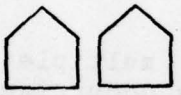
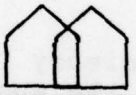
pattern(s). The referenced field may not be binary or coordinate mode. If the variable set is referenced, qualification is at the record level rather than at set level.

CIR, CIRCLE -- Are relational operators used to test if a point falls within a circle specified as a data value. The point being tested must be stored in an internal coordinate mode field or group.

OVP, OVERLAP, OVERLAPS -- Are relational operators used to test if a point, or polygon overlaps a data value polygon. A polygon is specified by any number of coordinates from one to eight. A one-coordinate polygon is a point, a two-coordinate polygon is a line, and a three-or-more coordinate polygon is an enclosed area polygon. No polygon may overlap the Greenwich Meridian (Ø degrees longitude), which is a two-coordinate polygon. The polygon being tested must be stored in an internal coordinate mode field or group. Table 2 illustrates the true and false results of the four permissible OVP relations.

Note that only the CIRCLE search and OVERLAP relational operators may be used with coordinate fields.

Table 2

PERMISSIBLE OVP RELATIONS		
	No Overlap	Overlap
Point versus Area		
Line versus Area		
Line versus Line		
Area versus Area		

The enclosed area polygon specified to the OVP operator may be concave or convex and must have the coordinates of the vertices defined in a counterclockwise order. Note that only the above four combinations of polygons may be tested for overlapping.

Data values -- The number and type required vary with the relational operator and are described in the following paragraphs.

Self-defining term -- Is data whose value is inherent in the term. Examples of valid self-defining terms are: 15Ø, F-111, ABC.

Literal -- Is data whose value is enclosed in quotation marks. Any term which contains a special symbol other than a dash or a dollar sign must be enclosed in quotation marks; i.e., introduced as a literal. A literal used as a single data value may contain all but the following combinations within the literal:

')

```
'blank
'.
',
```

A literal used as a subfield of a term, e.g., BT 'NEW LONDON'/'NEW YORK', may contain all but the following character combinations within its delimiters:

```
')
'blank
'.
',
```

Note: For efficiency when using multiple data values specify the most frequently occurring value first.

Indirect-address literal -- Is the name of a field, the contents of which will be used as a data value. It may not be a variable length field or the variable set. The field name must be preceded with an ampersand (&). When this capability is used, the contents of the field in the data base are used to make the comparison. All fields referenced in one clause must belong to the fixed set or to the same periodic set. Indirect-address literal usage will nullify Index Processing for the clause.

System-provided work areas -- Have the following attributes: There are five areas available for use. They are four bytes in length and may be referenced using system names WORK1 through WORK5. A work area can be designated as a parameter to the Function Operator, as a value in a condition clause or as a field in the sort key. Note, however, that the only means for placing data in a work area or resetting a work area is through use of the Function Operator and that processing of a query is in set sequence. There will be a set of work areas per IF or FURTHER statement.

Within a conditional (IF or FURTHER) statement, work areas containing the function-generated data are output on the QRT for each qualifying fixed set and/or periodic

subset. All work areas up to and including the highest referenced within the conditional statement will be output; i.e., if WORK4 was referenced, WORK1 through WORK4 will be output on the QRT. The work areas will be output to the QRT for all sets and qualifying subsets within the conditional statement. This may result in data not wanted for output being placed on the QRT. Obviously, only those work areas required for display or processing by the Output Processor should be referenced in the user's Report Instruction Table or QUIP query. The amount of work area data written on the QRT can be controlled through careful use of the WORK designation and with the SELECT statement (see subsection 2.4.3.2). If only one work area is required, then WORK1 should be used. If two work areas are used, WORK1 and WORK2 should be used, etc. If only data from a specific set is to ultimately be reported, then only that set should be selected for output to the QDF with QRT pointers generated accordingly.

Coordinate/radius (required for CIR) -- Is a data value composed of a coordinate in 11- or 15-character format, and a radius of 1 to 5 digits;

```
DDMMXDDDMMY/XXXXX
DDMMSSXDDDMMSSY/XXXXX
```

where

DD	=	latitude in degrees
MM	=	minutes in degrees
SS	=	seconds of degrees
X	=	N for North and S for South
DDD	=	longitude in degrees
Y	=	E for east and W for west
/	=	subfield separation symbol
XXXXX	=	radius in nautical miles not to exceed 10800.

The coordinate will be converted automatically to internal format by a standard system subroutine (UTCORDI).

Coordinate/coordinate... (required for OVP) -- is a data value composed of two or more coordinates separated by

slashes and describes a point, line, or polygon. Vertices must be listed in counterclockwise order.

Note: OVP and CIR are the only relational operators which may be used with coordinate fields defined as 11 or more characters in length. OVP and CIP cannot be used to test coordinate data stored in other than coordinate mode fields and groups.

When the number of coordinates being specified exceeds the boundaries of a card (cc 1-71), the following convention must be followed:

```
IF POINT OVP coord/coord/coord        (1st card)
   coord/coord.                       (2nd card)
```

A slash may not follow the last coordinate specified in the first card. The second card may begin in any column (1-71).

Coordinate fields may be used with relational operators other than CIR and OVP if they are defined as 5-, 6-, 7-, or 8-character format; i.e.,

```
DDMMX          (Latitude)
DDDMMY         (Longitude)
DDMMSSX        (Latitude)
DDDMMSSY       (Longitude)
```

Comparisons of GT, LT, etc., are made based on the internal coordinate value. For latitude, the smallest value is at the South Pole (9ØØØØØS); and the highest value is at the North Pole (9ØØØØØN). For longitude, the lowest value is ØØØØØØØE, with value increasing around the world until the highest value is reached at ØØØØØØØW.

Note: Coordinate fields are initialized by FM to a value of 9ØØØØØS for latitude and ØØØØØØØE for longitude.

Coordinate fields which do not contain user data can be retrieved as follows:

a. If coordinate fields have been defined as 5, 6, 7, or 8 positions, use the appropriate clause

```
IF LAT EQ 9ØØØS.
IF LONG EQ ØØØØØE.
        or
IF LAT EQ 9ØØØØØS.
IF LONG EQ ØØØØØØØE.
```

where LAT and LONG represent the names of coordinate fields.

b. With coordinate fields of any length, a FUNCTION subroutine clause can be used, as follows:

```
IF FUNCTION CDZERO POINT.
```

where POINT is a coordinate field and CDZERO is a subroutine to determine if the field contains all binary zeros.

The data value portion of a clause indicates what values to test for in the field preceding the relational operator. If a data value contains any special characters, except dollar sign or hyphen, that data value must be enclosed in single quotes. Unless the relational operator is CONTAINS, the data values will be padded with blanks in the rightmost positions if the FFT specified the field as alphabetic. If the field is numeric, the leftmost positions will be padded with zeros to the field size specified by the FFT. Data values longer than the field size specified by the FFT are invalid. If the field is coordinate, all 11 or 15 characters must be specified.

The use of a dollar sign ($) in a data value parameter indicates partial-field notation and causes that character position to be ignored when compared to its corresponding character in the data base field. Dollar signs must not be used in data values tested by the relational operators CONTAINS, CIR and OVP. Dollar signs in a data value will nullify Index Processing for the clause. If dollar signs appear in a data value, that value may not be converted by a subroutine.

An example of the use of the dollar signs in a data value is:

IF HIER EQ AB$$.

If the data base field HIER contains ABGO, a true condition will result. Note that only one dollar sign in the data value would have been sufficient since the parameter would be padded with dollar signs in the rightmost position; i.e., IF HIER EQ AB$. would be valid. The dollar sign may appear at the high- or low-order positions in the data value but may not be embedded within the data value. For example, AB would be legal, but AB would be illegal.

When using multiple data values with the dollar sign capability, all data values must have the same pattern. For example, IF FIELDA EQ AB$, AC$. would be legal, but IF FIELDA EQ AB, AC. would not.

Multiple data values may be used only with the EQ, NE, CONTAINS, and BT relational operator. The following are conditional statements with multiple data values:

IF MEQPT NE A-4C, HU-16A, C-54P.

IF PLAN BT 30$/40$, 60$/65$.

The first condition is true if some value other than A-4C, HU-16A, or C-54P is contained in the major equipment field. The second condition is true if the plan field lies within either of the two ranges.

When using the NE relational operator or the modifier NOT with the EQ, BT, or CONTAINS operators and multiple data values, there is an implied AND-type logical connector between each data value. With the EQ, BT, and CONTAINS operators when the modifier NOT is not used, there is an implied OR-type logical connector between the data values.

Retrieval Logic -- An IF statement may be composed of several clauses linked by the logical connectors AND or OR to form logical conditions. All clauses joined by AND must be satisfied for the logical block to qualify on this

clause. Clauses joined by AND are evaluated together; then the OR logic is performed on the results. To modify this order, parentheses may be used. The clauses within parentheses are evaluated independently, and where nested parentheses are used, the expression contained in the most deeply nested parentheses (that is the innermost pair of parentheses) is evaluated first. Successively higher levels are evaluated until the truth factor for the entire statement has been determined. However, flagging is done from left to right; this is discussed under Periodic Set Logic.

The following examples illustrate the Boolean logic of retrieval. All the conditions for which the statement is true are illustrated. T is used for true and F for false. Assume that the clauses represented by A, B, C, D reference fixed set fields only.

Example 1

A AND B	
T	T

A OR B	
T	T
F	T
T	F

Comment: This example illustrates simple AND and OR logic, showing that both conditions A and condition B must be true in the AND case, but that the truth of either or both is sufficient for OR.

Example 2

A OR B AND C		
T	T	T
T	F	F
F	T	T
T	T	F
T	F	T

(A OR B) AND C		
T	T	T
F	T	T
T	F	T

Comment: In RASP, implied parentheses surround each AND group; thus the logic of the leftmost table stated explicitly would be A OR (B AND C). In the example at the right, the terms were explicitly grouped by enclosing the terms A OR B in parentheses. Other uses of parentheses are given in examples 3 and 4.

Example 3

(A AND B)		OR	(C AND D)	
T	T		F	F
T	T		T	F
T	T		F	T
T	T		T	T
F	F		T	T
F	T		T	T
T	F		T	T

Comment: The explicit form of parentheses is illustrated by the statement shown above. The table would be the same if the parentheses were omitted.

Example 4

A AND (B		OR	C) AND D	
T	T		T	T
T	T		F	T
T	F		T	T

Comment: Compare this example with the preceding illustration. Here, parentheses have been used to regroup the elements of the expression and the logic has been altered significantly.

Example 5

A OR B		OR	C AND D	
T	F		F	F
T	T		F	F
T	T		T	F
T	T		T	F
F	T		F	T
F	T		F	T
F	F		F	T
T	F		T	T

Comment: This illustrates compound OR logic with a single AND condition.

Periodic Set Logic -- The following examples illustrate the logic of retrieval applied to the periodic set(s). The left-to-right logic scan shows up when periodic fields are referenced. The labels FX, P11, P12, P21, etc., are used to indicate a fixed field, a periodic field from periodic Set 1, and periodic field from periodic set 2. Successive terms against the same periodic set are evaluated one subset at a time if they are within the same parentheses level. If the subset qualifies within the terms of the expression, it is flagged with an asterisk (*). The fixed field condition is always assumed to be true in these examples.

Example 5

A	OR	B		OR		C	AND	D
T		F				F		F
T		T				F		F
T		T				T		F
T		T				F		T
T		T				T		T
F		T				F		F
T		F				T		F
F		T				T		F
F		T				F		T
F		T				T		T
T		F				F		T
F		F				T		T
T		F				T		T

Comment: This illustrates compound OR logic with a single AND condition.

Periodic Set Logic -- The following examples illustrate the logic of retrieval applied to the periodic set(s). The left-to-right logic scan shows up when periodic fields are referenced. The labels FX, P11, P12, P21, etc., are used to indicate a fixed field, a periodic field from Periodic Set 1, a second periodic field from Periodic Set 1, and periodic field from Periodic Set 2. Successive terms against the same periodic set are evaluated one subset at a time if they are within the same parentheses level. If the subset qualifies within the terms of the expression, it is flagged with an asterisk (*). The fixed field condition is always assumed to be true in these examples.

Example 1

IF FX AND P11.

Subset	P11	Flags
1	T	*
2	T	*
3	F	
4	T	*

Comment: This logical block will be selected because both the fixed and at least one periodic subset qualify.

Example 2

IF P11 OR P12.

Subset	P11	P12	Flags
1	T	F	*
2	T	T	*
3	F	T	*
4	F	F	

Comment: Because P11 is the leftmost term of the expression, it is evaluated first. If it qualifies, the subset P12 does not need to be considered and scanning proceeds with the next subset. If P11 fails to qualify the subset, the next term in the expression, P12 in this case, is evaluated. All qualified subsets are flagged.

Example 3A

IF FX AND P11 OR P12.

Subset	P11	P12	Flags
1	T	T	*
2	T	F	*
3	F	T	
4	F	F	

Comment: The implied parentheses around FX and P11 prohibit flagging of the P12 field even though it is in the same periodic set as P11. The P12 field will be flagged only if the expression formed by the first two terms (FX and P11) is not satisfied within the periodic set, but P12 is satisfied.

Example 3B (with SUBSCAN option)

IF FX AND P11 OR P12.

Subset	P11	P12	Flags
1	T	T	*
2	T	F	*
3	F	T	*
4	F	F	

Comment: The implied parentheses have no effect in this case, and all subsets satisfying the condition are flagged.

Example 4

IF FX AND (P11 OR P12).

Subset	P11	P12	Flags
1	T	T	*
2	T	F	*
3	F	T	*
4	F	F	

Comment: Use of the parentheses regroups the term of the expression and allows normal set scanning. All terms from the same set are evaluated concurrently if the terms are not grouped by actual or implied parentheses and if the terms are contiguous in the input stream.

Example 5

IF FX AND P11 AND P12.

Subset	P11	P12	Flags
1	T	T	*
2	T	F	
3	F	T	
4	F	F	

Comment: AND successive periodics from the same set imply the requirement for satisfaction from the same subset.

Example 6

IF FX AND P11 AND P21.

Subset	(Set 1) P11	Flag	(Set 2) P21	Flag
1	T	*	T	*
2	T	*	F	
3	F		T	*
4	F		F	

Comment: This illustrates the rule that each set is scanned and flagged independently. When all conditions of the statement have been satisfied, the record qualifies.

Example 7

IF FX AND P11 AND ANY P12.

Subset	P11	Flags After Scan on P11	P12	Flags After Scan on P12
1	T	*	T	*
2	T	*	F	
3	F		T	*
4	F		F	

Comment: The net results are, of course, flags on subsets 1, 2, and 3. Use of the ANY modifier forces the system to consider the P12 as if it queried a different set than P11; i.e., flagging is performed on other conditions of the subset independently.

Example 8A

IF P11 OR ANY P12.

Logical Block	Subset	P11	Flags After Scan on P11	P12	Flags After Scan on P12
1	1	T	*	F	
1	2	T	*	T	
1	3	F		T	
1	4	F		F	
2	1	F		F	
2	2	F		T	*

Comment: This illustrates that IF P11 OR ANY P12 is different from IF P11 OR P12. The latter case is shown in example 2. Use of the modifier ANY forces the system to treat P12 as if it queried a different set. The system examines the leftmost term of the expression (which in this case is P11). If the first term of the expression qualifies the logical block, the ANY term is never examined. By this rule, subset 3 in logical block 1 is not flagged. If the first term fails to qualify at least one subset, the ANY term is evaluated. Those subsets qualified by the ANY term will be flagged. In this example, P12 qualifies subset 2 in logical block 2; therefore, the subset is flagged.

Example 8B (with SUBSCAN option)

IF P11 OR ANY P12.

Logical Block	Subset	P11	Flags After Scan on P11	P12	Flags After Scan on P12
1	1	T	*	F	
1	2	T	*	T	
1	3	F		T	*
1	4	F		F	
2	1	F		F	
2	2	F		T	*

Comment: The use of SUBSCAN forces normal subset scanning and becomes exactly the same as example 2.

Example 9

IF FX AND P11 AND P21 AND P12 AND P22.

	(Set 1)			(Set 2)		
Subset	P11	P12	Flag	P21	P22	Flag
1	T	T	*	T	T	*
2	T	F	*	T	F	*
3	F	T	*	F	T	*
4	F	F		F	F	

Comment: Crossing set boundaries causes the system to evaluate the terms of the expression independently. The conditional is evaluated as if it were expressed in either of the following formats:

IF (FX AND P11)

AND (P21 AND P12)

AND P22.

or

IF FX AND ANY P11

AND ANY P21 AND ANY P12

AND ANY P22.

Three of the four ways in which the system can be forced to evaluate the terms of an expression independent from the other terms in the subset have been illustrated. These are: crossing set boundaries, the use of parentheses (explicit or

implied), and the user of the ANY modifier. The fourth way will be defined and illustrated in the discussion of the FURTHER statement, section 2.4.2.4.

The follcwing examples illustrate various forms of the IF statement.

Example 1

```
IF MEQPT EQ F-1Ø5 AND

MEPSD EQ &MEORC.
```

Comment: This request is for all units having fully operational F-1Ø5s as their major equipment. The program selects the appropriate records by checking the number contained in the major equipment possessed field against the number in the number operational field.

Example 2

```
IF HIER 3/3 EQ 'Ø'.
```

Comment: The user may specify a partial field by following the field/group mnemonic with the high- and low-order positions of the partial field. The high- and low-order positions are separated by a slash.

Example 3

```
IF SERV EQ ARMY,NAVY.
```

Comment: In most files there are some fields which contain coded data values. The RASP retrieval does not require that coded data values be stated in their coded form unless the relational operator being used is CONTAINS. The retrieval parameters will be converted to their coded form if a subroutine or table was specified by the file format table. For example, the values for ARMY and NAVY might be carried as W and N in the SERV field. The retrieval can be written

as above if the Service Code Conversion Table RCMDS is specified in the file format table for input conversion.

Example 4

```
IF SERV #RCMDS# EQ ARMY,NAVY.
```

Comment: The RASP component can also be told to convert data values by a particular subroutine. By preceding or following the relational operator with the name of a subroutine enclosed in pound signs (#) the conversion can be forced. In the preceding example, RASP is forced to convert the data values by table RCMDS, even though RCMDS may or may not have been specified by the FFT.

When Index Processing is used, the data values in the retrieval statements are compared with the field values in the Index Data Set. The values may be compared as they are submitted, they may be converted from an external format as specified in the FFT, or they may be converted to an internal data file format by a subroutine. The values may be converted from external format to a file format as specified in the FFT Element Descriptor Records, and in addition converted from the file format to an index format by a conversion subroutine specified at Index Specification time; or, the values may merely be converted to an index format by a conversion subroutine without first being converted to an intermediate file format. Conversion by subroutines specified during File Structuring or Index Specification will be automatic.

It is possible to produce multiple values from one input entry. That is, a conversion subroutine may generate search value "synonyms." If multiple values are produced, the relational operator must be EQUAL (or a valid synonym). If not, Index Processing will be negated for the clause. For each entry, Index Processing will construct a clause (field EQUAL value) and connect them by "OR".

The user may specify the use of a subroutine by enclosing it in pound signs. This explicit subroutine conversion refers to external-to-internal data file format

conversion only. Conversion from data file format to Index Data Set format cannot be overridden, either at file maintenance or retrieval times.

Example 5

```
IF SERV ## EQ A, N.
```

Comment: External-to-internal (but not internal to Index Data Set format) subroutine conversion may be suppressed by writing the double pound sign (##) either before or after the relational operator. Data values will be compared as they are submitted with data file values, but will be converted by the routine indicated (optionally) as Index Specification for comparison with Index Data Set values.

Example 6

```
IF SERV EQ #RCMDS# ARMY,NAVY

#SPCSV# 'COAST GUARD'.
```

Comment: When multiple data values are used, it is possible to specify a maximum of two conversion routines for data values in the same clause. These may consist of the combinations formed by any two of the following: default to the FFT, special conversion routine, and suppression of the subroutine conversion. When a particular mode of conversion is specified, it applies to all subsequent values of the clause until it is replaced by another conversion routine.

Example 7

```
IF RADTG BT 68Ø3151700 AND

68Ø4012300.
```

Comment: In this example, one BT (BETWEEN) operator is used to retrieve the record for those units which will attain expected combat readiness between (and including) 1700 hours

on 15 March 1968 and 2300 hours on 1 April 1968. The two values may be linked by the word AND or by a slash.

Example 8

IF POINT CIR 4851N00220E/225.

Comment: The CIR (Circle Search) relational operator is used to test if a point falls within a radius of 225 nautical miles of Paris, France.

The coordinate may be expressed in 11- or 15-character format.

Example 9

IF MEPNT OVP

4900N00650E/4940N00615E/4945N00220E/4830N00220E.

Comment: This illustrates the use of the polygon search capability to locate units whose major equipment is deployed within a selected area. MEPNT is a periodic field which contains the deployment coordinates in 11-character format.

Example 10

IF ATACH EQ LANT, EUR, PAC, SAC

AND MECL NE AK, MTA, MTY.

Comment: This condition is true if the unit is attached to the Atlantic, European, Pacific, or SAC Commands, and it does not possess major equipment in the coded categories AK, MTA, or MTY. Use of multiple data values instead of multiple conditions makes the retrieval more efficient.

Example 11

```
IF UNTYP EQ 24VC$, 24VS$.
```

is valid but the following is not permitted

```
IF UNTYP EQ 24V$C, 24VS$.
```

Comment: When using multiple data values and the dollar sign ($) feature, all data values must have the same pattern. Moreover, the data values must not have an embedded dollar sign. This feature is not applicable for the CONTAINS relational operator.

Example 12

```
IF ATACH EQ EUR
AND MECAP EQ D AND METRY EQ GW
AND ANY METRY EQ IT.
```

Comment: This conditional is designed to retrieve units attached to the European Command which have particular weapons in West Germany and which also have any major equipment in Italy. Since all terms of the expression are in an AND relationship, all terms must be satisfied. ATACH is in the fixed set while MECAP and METRY are both in Periodic Set 1.

Example 13

```
IF ATACH EQ SAC AND
((GEPOL EQ NA AND CNTRY NE CN AND MEDEP EQ 'M')
OR (MEPOL EQ NA AND METRY NE CN AND MEDEP EQ'M'))
AND UIC  EQ J$, M$.
```

Comment: This illustrates the use of nested parentheses. The conditional requests logical blocks for units whose status is reported by the Strategic Air Command. The unit must be located in the North American Continent, but not in Canada, and have no major equipment deployed; or the geo-political area in which the unit may be deployed must be in the North American Continent, but not in Canada, and it must not have major equipment deployed currently. In either case, the unit identification code must begin with the letter J or M.

2.4.2.3.2 Condition Clauses and Secondary Indexing

Sections 1.1, Secondary Indexing, and 1.3, System Flow, described under what conditions Index Processing is invoked by RASP. This section explains how, once invoked, Index Processing determines if the indexing information about the file can be utilized to direct RASP in accessing the data records.

Index Processing bases its decision on the presence of an indexed field in at least one "must-satisfy" clause of the retrieval that does not use the CONTAINS relational operator and is not an ABSENT conditional clause. A must-satisfy clause is defined as a clause whose condition must be true in the data record in order for that record to qualify for retrieval. Whether a clause is in the must-satisfy state or not depends on how the retrieval is written. Therefore, the decision to utilize index information or not is always based on the dynamic structure of a retrieval at run time.

The basic criterion in deciding whether a clause is must-satisfy or not is if the clause (or string or group of clauses) is connected to a clause of equivalent logical level by AND. Connection by AND ensures that the clause condition must be satisfied in order to retrieve the record. Connection by OR gives the processor a choice of terms by which to qualify the record. Example:

IF CNTRY EQUAL US.

IF CNTRY EQUAL US AND MEQPT EQUAL TANK.

Comment: In each of these statements, the clause CNTRY EQUAL US is a must-satisfy clause, because this condition must be true in order to retrieve the record.

Contrast the above examples with this:

IF CNTRY EQUAL US OR SERV EQUAL ARMY.

Comment: In this case, CNTRY EQUAL US is not in the must-satisfy state. If this condition is not true, RASP can still qualify the record if SERV is equal to ARMY, irrespective of the field CNTRY contents.

2.4.2.3.2.1 TEST360 Indexes

Refer to Volume II, File Structuring, Section 3.3, Sample File Structure Job, for a description of the TEST360 file. Note that SERV and CNTRY are fixed-field indexes, and MEDDT and METRY are Periodic Set 1 indexes. PLRT, a periodic field in Set 3, and SPCODE, a periodic field in Set 5, complete the list of indexes.

2.4.2.3.2.2 Examples Using Indexed Fields

In these examples, the operator is EQUAL, however, any valid operator except CONTAINS or ABSENT could be used. Any index, fixed or periodic, may be used. Qualification is at the data record level, not at set level.

Example 1.

IF CNTRY EQ GY.

Comment: This example shows a retrieval with one clause. Only the records whose field CNTRY equals GY will be present as candidates. In this example, of course, all candidates are also qualifiers.

Example 2.

IF PERS GT 10000 AND CNTRY EQ UK.

Comment: Each clause is a must-satisfy clause. That is, PERS must have a value greater than 10000 and in addition the country must equal UK. As CNTRY is indexed, the search for qualifying records can be restricted to those records whose field CNTRY contains the value UK.

Only these records would be searched in detail to find the qualifying records; i.e., records whose field PERS contains a value greater than 10000.

Example 3.

IF CNTRY EQ GY OR CNTRY EQ FR.

Comment: The search would be restricted to records containing GY or FR in the field CNTRY.

Example 4.

IF CNTRY EQ GY OR PERS GT 10000 AND CNTRY EQ UK.

Comment: Records containing GY as the CNTRY, or records containing UK as the CNTRY and more than 10000 in PERS would qualify. Index Processing would restrict the search to records carrying GY or UK in CNTRY.

Example 5.

IF (CNTRY EQ GY OR CNTRY EQ UK) AND PERS GT 10000.

Comment: Although the logic of this example would restrict qualifying records to those with GY or UK in CNTRY and PERS greater than 10000, Index Processing would present the same records as in example 4.

Example 6.

IF (SERV EQ ARMY AND UNRDY EQ A) OR (REQPT EQ A AND METRY EQ US).

Comment: In each parenthetical group an indexed field is referenced in a must-satisfy clause. Since the qualifying

records could contain either combination of values as expressed in each group, retrieval must search all records in which SERV equals ARMY or METRY equals US. Both could of course be present in one record, but Index Processing would only be concerned with the either/or case. All other records would be excluded from the detailed search.

Example 7.

IF UNTYY NE ARTL AND (SERV EQ ARMY OR METRY EQ US) AND UNTYZ EQ X.

Comment: The logic of this example would restrict the qualifying records to those whose UNTYY field was not ARTL, and whose UNTYZ field was X. In addition, the SERV field must equal ARMY or the METRY field must equal US. As the group is a must-satisfy group, one clause or the other must be true in order for the data record to qualify. Therefore, Index Processing would determine that only records whose field SERV contained ARMY or whose field METRY carried US could possibly qualify. In this manner, all other records would be eliminated from the detailed search.

2.4.2.3.3 ABSENT Conditional Clause

Whereas the standard conditional clause described in section 2.4.2.3.1 provides record qualification based on the contents or at least the existence of certain fields in the records, the ABSENT conditional clause provides the capability to qualify records strictly on the basis of the absence of a specific periodic or variable set. This clause causes the referenced set, identified by a field or group name from that set, to be examined. If no subset exists for the record, the condition is true. This clause must be used in conjunction with field names or group names from periodic or variable sets. The field name or group name used serves as an indication of the particular set being referenced. Secondary indexing is not applicable to this clause type.

The format of the ABSENT conditional clause is:

field name
group name [NOT] ABSENT

where

Field name, group name(required) - is either a field name or a group name from the periodic or variable set which is to be examined.

NOT(optional) - reverses the truth value of the ABSENT operator.

ABSENT(required) - is the operator which identifies the clause and is used to examine the set referenced to see if it exists in the record.

Note: Sorting of records which qualify by means of the ABSENT conditional clause must not involve fields from the set in question. See section 2.4.2.3.3.1,

Example 1

IF LOC EQ PARIS AND MEQPT ABSENT.

A record would qualify if the value of the fixed set field LOC were PARIS and there were no subsets in periodic set one. Periodic set 1 is referenced by specification of the field name MEQPT which is defined in the FFT for the file as a field from set 1.

Example 2

IF MEPSD NOT ABSENT.

This request is for all records where there is at least one occurrence of a subset in periodic set 1. Qualification with the ABSENT operator is at the record level, and flagging of the subsets is not accomplished with the ABSENT conditional clause.

2.4.2.3.3.1 ABSENT/SORT Restrictions

Care must be exercised when attempting to sort on periodic fields from set involved in an ABSENT clause. The ABSENT clause qualifies records based on the absence of a

periodic set and; therefore no subsets are flagged. However, sorting on perodic fields requires that at least one subset be flagged. These two operators, when combined, cancel each other and cause those records which qualified based on ABSENT logic to be eliminated from the QRT. This situation might arise in a case where records are to be retrieved regardless of the presence of a periodic set; yet, if the set is present sorting is to take place on a field in the set. The following is an example of combining ABSENT with periodic sorting.

```
IF LOC EQ PARIS
AND MEQPT EQ $MEQPT.
SORT UIC, MEQPT.
IF LOC EQ PARIS
AND MEQPT ABSENT.
SORT UIC, '    '.
```

In this example it is desired to retrieve all records which have the value PARIS in the field LOC. Additionally, if the periodic set containing the field MEQPT is present, the field is to be used in the SORT statement. The first IF/SORT statements retrieve the records for PARIS with MEQPT present. The second IF/SORT statements retrieve those PARIS records where MEQPT is not present. MEQPT's position in the sort key if filled with a literal containing blanks. (For a description on the SORT statement see section 2.4.3.1).

2.4.2.3.4 FUNCTION Operator

The FUNCTION Operator is a conditional clause within the IF or FURTHER conditional statement. It is normally preceded and followed by one of the logical connectors in the statement. The use of the FUNCTION Operator is designated by the keyword FUNCTION, followed by the name of the function (the user's subroutine which may be enclosed in pound (#) signs). All terms following the function name and preceding the next logical connector or the end of the statement are considered as part of the parameter list to the function subroutine. The parameters may consist of field names, literals and system-provided work areas.

Field names must be designated by prefixing the field name with an ampersand character. Within a parameter list, fields may be referenced which, belong to the fixed set and one periodic set, but not to two different periodic sets.

System-provided work areas will be designated by the system labels WORK1 through WORK5. No distinction is made as to whether a work area is input to or output from the subroutine.

Any item in the parameter list which is not a system work area or is not prefaced with an ampersand character is processed as a literal. Literals containing embedded blanks or commas must be enclosed in quotes, and literals prefaced with a plus or minus sign (+ or -) must also be enclosed in quotes.

Partial-field notation or user subroutine conversion will not be allowed with items in the parameter list.

Example

```
IF...AND FUNCTION GCD,&POINT,6Ø75N12323E,9ØØ,GT,
WORK1 OR...
```

Comment: Assume that GCD is a user-written subroutine which, given two latitude/longitude points, will compute the distance between the two points and compare it to a given distance. It then will set the true/false indicator according to the given condition. The computed distance will be placed in a designated work area. The order of parameters is Point 1, Point 2, distance, qualifier and work area.

In this example, Point 1 is a data file field POINT. Point 2 is a user-designated point 6Ø75N12535E. Note that for coordinates in a parameter list, conversion to internal form is not automatic. The designated distance is 9ØØ and to qualify, the computed distance must satisfy the condition GT. Note here that GT is a requirement of the subroutine GCD. To RASP, GT is a literal value to be entered as a parameter to GCD. The computed distance would be placed in the system-provided work area WORK1.

RASP only provides an interface to the user-written function subroutine. The number of parameters, the order of parameters, whether parameters can be omitted, and invalid parameters (to the subroutine) are all functions of the subroutine. RASP provides the data and control for the subroutine in the order in which they appear in the query. If the subroutine will allow a variable number of parameters, omission within a parameter list cannot be denoted by use of the double comma. To omit a parameter within the list which is positional in nature, an omit code which is recognized by the subroutine must be used.

There are several rules or restrictions which the user must follow to use the FUNCTION Operator. Basically, the FUNCTION Operator and any correlation between subroutine requirements and designated input are the user's responsibility. If the following rules are heeded, there should be little problem using the FUNCTION Operator.

a. A double comma cannot be used to indicate omission of a parameter.

b. Parameters must be designated in the order prescribed by the subroutine.

c. Literal values in the parameter list are subroutine related and not data field related; thus, there will not be any automatic conversion by subroutines of literals on input.

d. A literal containing embedded blanks or commas, or a literal prefaced with a plus or minus sign, must be enclosed in quotes.

e. A literal containing any nonnumeric character will be passed to the function subroutine as EBCDIC characters; this includes signed numeric values also, because the plus and minus signs are not interpreted as numeric characters.

f. A literal containing all numeric characters will be converted to a binary value prior to being passed to the function subroutine. Enclosure of an all

numeric value in quotes will not prevent that value from being converted to binary; i.e., a literal must contain a nonnumeric character to be passed to the function subroutine as an alpha literal.

g. Data file field names must be prefixed with an ampersand.

h. The parameter list to the FUNCTION Operator subroutine cannot contain the word AND or OR.

2.4.2.3.5 FUNCTION Operator Usage by Index Processing Analyzer Routines

The FUNCTION operator parameter string will be examined to determine if a data file field, or fields (a data field name must be prefixed by an ampersand), is present. If so, and any field so referenced is indexed, Index Processing will be invoked. If no field name is given in the FUNCTION operator parameter string, or no field is an index field, index usage will not be possible for that clause. Only one indexed field per function clause (the first one encountered) can trigger index usage. Any other indexed fields are treated as nonindexed fields in the analysis.

The indexed field used in a FUNCTION operator clause must have an analyzer routine specified for it. An analyzer routine is designated for a field in an Index Specification run (reference Volume II. File Structuring). The purpose of the analyzer routine is to analyze the FUNCTION operator parameters for Index Processing. The analyzer routine will return a value (or values) which Index Processing will use as arguments when searching the field's index information in the Index Data Set. This search is conducted in an "equal" relationship only.

Index Processing cannot determine how the FUNCTION operator parameters interrelate; this must be done by the analyzer routine. Therefore, if no analyzer routine is specified for any index field, Index Processing will be negated for the entire FUNCTION operator clause.

Index Processing only provides an interface to the user-written analyzer subroutines, exactly as RASP provides the interface to a Function subroutine. All conditions, rules, and restrictions regarding Function subroutines, as described in Section 2.4.2.3.4, FUNCTION Operator, above, apply to an analyzer subroutine. Section 2.8, Writing an Analyzer Subroutine, provides information for calling sequence and parameters needed.

2.4.2.3.6 Conversion Subroutine/Tables

The purpose of a conversion routine is to convert a data file value to the format desired for the Index Data Set. This function is designated during Index Specification and is performed during Index Maintenance and Index Processing. It cannot be overridden.

A conversion routine is written following the normal subroutine conventions, not those for an analyzer routine. A return code of 'S' indicates successful execution; a return code of 'M' indicates an unsuccessful execution. Any other code will cause the Secondary Indexing routines to pass control back to the routine (with a fresh copy of the parameter string) for further conversion. This allows a conversion subroutine to generate synonyms for data file values to reside on the Index Data Set. The routine must save its own registers and status before returning control. It indicates completion (i.e., do not return any more) with a code of 'S'.

2.4.2.4 FURTHER

The FURTHER statement is optional. It is an extension of the IF statement logic and is connected to it by an implicit AND or an explicit AND or OR. It allows the user to state the basic conditions for retrieval in the IF statement and to state additional conditions in a FURTHER statement. Any other FURTHER statements before the next IF statement imply the conditions imposed by the first IF statement but not those expressed in the intervening FURTHER statement.

The format of the FURTHER statement is the same as the IF statement except that it begins with the statement identifier FURTHER instead of IF, and the word FURTHER can be followed by the noise word IF or the logical connector AND or OR.

The following examples illustrate various uses of the FURTHER statement.

Example 1

```
IF UNLVL EQ SQ AND MEDEP EQ 0
     AND MEQPT NE 0
     AND MECL 1/1 EQ H, M.
SORT/1 ATACH, SERV, UNLVL, 'AA', MECL 1/1.
FURTHER IF FUTI NE ## '   '.
SORT/2 FUTI, SERV, UNLVL, 'AA', MECL 1/1.
```

Comment: This example illustrates the use of a single FURTHER statement. Records meeting the conditions of the basic IF statement will produce one group of sort keys. In addition, the Future Command field (FUTI), i.e., the Command to which the unit has a pending transfer, will be tested. Records meeting the combined IF and FURTHER conditions will produce a second group of sort keys.

Example 2

```
IF ATACH EQ EUR AND SERV EQ W
AND CNTRY EQ GY.
FURTHER AND UNTYP EQ 17HB$,
     17HC$, 17HD$, 17HE$,
     AND UNLVL EQ BN AND
```

```
     MEDEP EQ 0.
SORT 'A', UNLVL, UNTYP, CNTRY.
SELECT MEQPT/F.
FURTHER UNTYP EQ 17MB$, 17ME$,
     17MF$ AND UNLVL EQ BN.
SORT 'B', UNLVL, UNTYP, CNTRY.
SELECT MEQPT/F.
```

Comment: This example illustrates the use of the FURTHER statement to produce two groups of answers in the same answer set sorted by unit organization level, type, and the country to which it is assigned. A literal has been placed in the high-order position of each group of answers. The output components can test for changes in the high-order position of the sort key to condition such output actions as page eject, change in page labels, accumulated totals for each group, and an accumulated grand total.

Example 3

```
FILE TEST36Ø.
TITLE RVØ2R/68Ø3 TEFORR ADD.
IF ATACH EQ EUR AND SERV EQ J
AND UNLVL EQ WG,SQ.

FURTHER IF MEORN NE Ø.
SORT/1 CNTRY, LOC, UNC.
SELECT MEQPT/F.
```

```
        FURTHER IF MEORN EQ Ø.
        SORT/2 CNTRY, LOC, UIC.
        SELECT MEQPT/F.

        FURTHER IF MEORC NE Ø.
        SORT/3 CNTRY, LOC, UNIC.
        SELECT MEQPT/F.
        FURTHER IF MEORC EQ Ø.
        SORT/4 CNTRY, LOC, UIC.
        SELECT MEQPT/F.
```

Comment: This illustrates the capability of the system to retrieve, select, and sort periodic information into several answer groups according to classes of data in the file. In this case, all results will be tabulated with one RIT called the Test File Operational Readiness Report (TEFORR).

Example 4

```
    FILE TESTFIL.
    TITLE TEST1/1.
    IF ATACH EQ EUR AND SERV EQ J AND MEQPT EQ A-4C.
    SORT 'A' UIC LOC.
    FURTHER MEQPT EQ UH-1D.
    SORT 'B' UIC LOC.
    FURTHER MEQPT EQ B-52.
    SORT 'C' UIC LOC.
```

Comment: This example illustrates the use of the FURTHER statement to force independent evaluation of terms against the same set, as if they were against different sets. The condition, MEQPT EQ A-4C, in the parent IF statement is evaluated against subsets in Periodic Set 1 separately from the scan for the FURTHER conditions, MEQPT EQ UH-1D and MEQPT EQ B-52. Records possessing both A-4C and UH-1D MEQPT will produce sort keys with the literal 'B.' Records possessing both A-4C and B-52 MEQPT will produce sort keys with the literal 'C.'

2.4.3 Imperative Statements

The imperative statements within a retrieval are SORT and SELECT.

2.4.3.1 SORT

The SORT statement:

a. Permits the user to specify the fields and literals which are to be used to sequence the answer records.

b. Provides the medium for assigning labels to several different sequences of the same answer records.

c. Provides for the merging of answer records from more than one IF statement.

d. Provides for merging answer records from multiple files.

The format is:

```
[RIT-name]    SORT    [sort-number] [/action code]

   field-group name  [partial field]  subroutine name
        or                                  or
     literal                             #D#
```

Note: A literal may not be followed by partial field notation, subroutine conversion, or descending sort.

where

RIT-name (optional) -- Every SORT statement with a unique RIT-name will produce one entry in the Qualifying Record Table which uniquely identifies the answer set. This label, called the Retrieval ID, consists of the (required) 4-digit maximum retrieval number and the (optional) RIT-name. The RIT-name entry on each SORT statement allows the user to modify the Retrieval ID for all QRT entries produced by a particular SORT and SELECT pair. The RIT-name entry also designates the name of the RIT to be used to process the particular answer set. In this way, the conditional logic can be expressed once, the results sorted 'n' ways, and the output processed by different RITs.

The user may override the RIT specified in the SORT statement by providing a PUBLISH statement at OP execution time.

If no RIT name is provided with the SORT statement, the RIT name specified in the TITLE statement is used. If a RIT name does not appear in the TITLE statement, the user must specify a RIT name at OP execution time via the OP PUBLISH statement.

SORT (required) -- If the SORT statement is not used, records are presented to the output processors in file sequence. If the SORT statement is used, RASP will generate appropriate sort keys found in three record types in the Qualifying Record Table. The first of these records, the File Information Pointer (FIP), provides the output components with pointers to the File Control Records for files which have records in the QRT. One FIP record is generated for each file. The second record type, the Retrieval Equation Record (RER), provides the linkage between the user's definition of an answer set and a RASP-assigned retrieval code which identifies the answer set. It also contains a pointer to the physical location of the retrieval source statements placed in the Qualifying Data File. The output components use this pointer to obtain and print

the retrieval source statements which produced each answer set. RASP generates one RER record for each answer set; i.e., one record for each unique retrieval number and RIT-name combination. RASP also produces a third record type in the Qualifying Record Tables. This is the RASP/OP Answer Entry Record (ROAER). It consists principally of a sort key and a pointer to the fixed set and each periodic set or periodic subset selected. There is at least one entry for each Sort and Select pair under a qualifying IF or FURTHER in each batch of retrievals. At the completion of the retrieval, the QRT is sorted on information in the QRT sort keys so that it may be accessed sequentially by the output processors.

An understanding of the components of the ROAER sort key will enable the analyst to better understand the ordering procedure for data records. This sort key is shown schematically below.

RETRIEVAL CODE	TYPE CODE	ACTION CODE	USER-DESIGNATED SORT KEY	FILE CODE	CONTINUATION CODE

FORMAT OF THE
RASP/OP ANSWER ENTRY RECORD (ROAER)
SORT KEY

The high-order position of the ROAER sort key contains the two-byte binary retrieval code assigned by RASP for each unique combination of retrieval number and RIT name. The second sort key field indicates the record type within the QRT. The third sort key field contains the action code supplied following the SORT statement identifier. In function, this is identical to the use

of a literal in the high-order position of the user-designated sort key. The fourth sort field contains the sort fields designated by the user in each SORT statement. The length of the designated sort key may vary between answer sets but may not exceed 21Ø bytes. The designated fields specified in any one SORT statement may be from a fixed set or one flagged periodic set and may contain one or more literals. The fifth field contains a program-assigned sequence number which is used to maintain data extracted from the file in the sequence in which it appears in the file. It will not alter the designated sort because it is lowest in the sort key. This field contains the binary file code assigned by RASP and is used to link the file code to the file name contained in the File Information Pointer Record. The last field in the ROAER sort key contains a binary sequence number assigned by RASP to ensure correct sequence among all continuation records.

Succeeding elements of the ROAER record are structured as a variable length periodic subset consisting of three to eight fields. These fields contain the number of the set being referenced, the beginning of the set or subset in the QDF, and flagging and subset selection data, plus provision for additional program communication data.

Sort-number (optional) -- The sort-number, once used to preserve the order of consecutive SORT statements following a particular conditional statement, is no longer required. If used, it must be coded as a single digit in the range of Ø to 9 and affixed to the keyword SORT with no intervening blank.

/Action Code (optional) -- The action code parameter is a user-assigned 3-digit code in the range of Ø to 255 which is used to control the order in which the results of each pair of SORT and SELECT statements are presented to OP. The specified arrangement is accomplished by converting the operand to a one-byte binary number and placing the result in the sort field preceding the user-designated sort key and then sorting the answer records. In function, this feature is identical to the use of a literal in the high-order position of the user-

designated sort key. A literal may be used in combination with or in place of the action code.

A literal must be enclosed in single quotes but it may not contain any of the character combinations previously cited as illegal in a RASP literal. The maximum size of a literal may not exceed 21Ø characters.

Field-Group Name (required) -- At least one field name or one literal must appear in the operand of a SORT statement. The order in which the fields or literals appear indicates the order in which the user-designated sort key will be generated. There is no limit to the number of fixed or periodic fields which may be specified, but all periodic fields must be from the same set and that set must have been queried. The maximum number of characters which may be placed in the user-designated sort key field is 21Ø bytes. When the maximum allowable number of user-designated bytes is exceeded, the retrieval will be flagged as an error, and it will not be executed. When a WORK area (WORK1 through WORK5) is used in a SORT statement, it will receive a fixed or periodic set attribute according to the last usage of that specific WORK area in a FUNCTION operator clause. The set attribute remains the same for the WORK area until modified by usage in another FUNCTION operator clause.

Example:

```
1.  IF FUNCTION FSUB1 &MECLQ WORK4.

2.  SORT 'A' WORK4 MEPSD UIC.

3.  SORT 'B' WORK4 MEORC.

4.  IF FUNCTION FSUB2 &PLEAC WORK4.

5.  SORT 'C' WORK4 PLAN
```

Note: MECLQ, MEPSD, MEORC are in Periodic Set 1; PLEAC, PLAN are in Periodic Set 3.

In this example, WORK4 first receives a Periodic Set 1 attribute in statement 1. WORK4 can then be used as a Periodic Set 1 field (in combination with other Periodic Set 1 fields, fixed fields, or literals) in the following SORT statements. The set attribute is changed to Periodic Set 3 in statement 4. In statement 5, WORK4 acts as a Periodic Set 3 field, (outputting one sort key for each flagged subset in Periodic Set 3).

All WORK areas are independently edited in the above manner. Each is initialized to a fixed set attribute (once at the beginning of the RASP job, not for each TITLE). The set attribute is changed to periodic when a periodic field is contained in the FUNCTION clause prior to the WORK area reference.

Partial Field (optional) -- Partial field notation may follow a field or group name and designates that portion of the field or group that is to be inserted in the sort key. Partial field notation may not be applied to a literal, coordinate field, or binary field. It may be applied to fields being converted by an output subroutine.

Subroutine Name (optional) -- The RASP component can be directed to convert a parameter from its file form to another form before the value is placed in the sort key. This will not affect the form of the value carried in the Qualifying Data File. The name of the subroutine must follow the name of the field to be operated upon and must be enclosed by pound (#) symbols. RASP will determine the maximum output length produced by the subroutine and use this length in building the sort key. The subroutine so specified must be an output conversion subroutine with alpha or decimal output attributes.

#D# (optional) -- The RASP component can be directed to perform a descending sort on a field or group. This is done by specifying the descending sort flag, #D#, after the field or group name. If no descending sort flag is present, then ascending sort is assumed. There must be at least one blank or comma immediately preceding and following the descending sort flag. Partial field

notation may be used with the descending sort capability; however subroutine or table conversion may not be used. See example 10 for an illustration of descending sort specification. A descending sort is accomplished by causing the two's complement of the field to be inserted in the sort key. For example, if the field value was TANK, it would be represented in binary as:

1110 0011 1100 0001 1101 0101 1101 0010;

its two's complement would be:

0001 1100 0011 1110 0010 1010 0010 1101.

Thus, the displaying of this section of the sort key in either QUIP or OP will cause nonprintable characters (blank) to be displayed.

Example 1

```
SORT UIC, PLRT.
```

Comment: Illustrates a sort on Unit ID Code and Plan Response Time.

Example 2

```
SORT CNAM 1/12, LOC 1/16, HIER 1/1Ø.
```

Comment: This illustrates the use of partial field notation to select the high-order positions in various fields which are to participate in the sort.

Example 3

```
SORT CNTRY #CNTRYS#, LOC, SERV, TPNAM.
```

This illustrates the use of conversion routine to convert the major sort field from its 2-character encoded form to the full name.

Example 4

```
SORT CNTRY #CNTRYS#, LOC 1/12, SERV
 #OCMDS#, TPNAME 1/15.
```

Comment: This illustrates the use of output conversion routines and partial field notation as applied to different fields in the user-designated portion of the sort key.

Example 5

```
IF POINT CIR 43Ø6NØØ556E/75.
SORT/1 UNTLV, UNRDY, UNAME.
IF POINT CIR 49Ø9NØØ22ØE/6Ø.
SORT/2 UNTLV, UNRDY, UNAME.
```

Comment: This illustrates the use of the action code field to designate which conditional statement produced the answer record. The change in action code can be tested in the output processor and the result used to control a page break, additional spacing, header lines, body lines and similar format control actions.

Example 6

```
IF SERV EQ J, M, N, R, AND
UNLVL EQ SQ AND MEDPE EQ Ø
AND MEQPT NE Ø and MECL 1/1 EQ A.
SORT ATACH, SERV, UNLVL, MECL 1/2, 'AA'.
FURTHER FUTU NE ## '  '.
SORT2 ATACH, SERV, UNLVL, MECL 1/2, 'BB',
FUTU, UIC.
```

```
SORT ATACH, SERV, UNLVL, MECL 1/1, 'AAA'.
```

Comment: This illustrates several features of the RASP SORT statement. It illustrates the use of a literal in the low-order position of the sort key to provide a field which OP can test to distinguish among groups of answer records produced by the conditional logic. It illustrates the use of the optional sort suffix number to designate the number of consecutive SORT statements which apply to the preceding conditional logic. It also shows the use of partial-field notation, followed by a literal to force a particular arrangement using the low-order positions in the sort key.

Example 7

```
SORT ATACH, SERV, UNLVL, MECL, FUTU, UIC.

SØ1ØR SORT FUTU, SERV, UNLVL, MECL 1/1.
```

Comment: This illustrates the use of the RIT-name parameter field to assign a new Retrieval ID to the answer records produced by the second SORT statement.

Example 8

```
Line 1  TITLE SØ38R/11Ø3, TOPB38R.

Line 2  FILE INSTALN.

Line 3  NOTE ANY MILITARY AIRFIELD WITHIN

Line 4  3Ø MILES OF METZ.

Line 5  IF POINT CIR 49Ø9NØØ611E/3Ø.

Line 6  SORT/1 LOC.

Line 7  NOTE ANY MILITARY AIRFIELD WITHIN

Line 8  5Ø MILES OF TOULON.

Line 9  IF POINT CIR 43ØNØØ556E/5Ø.
```

RETRIEVAL AND SORT PROCESSOR (RASP)

```
Line 1Ø SORT/2 LOC.
Line 11 FILE TEST36Ø.
Line 12 NOTE ANY UNITS WITH DUAL WEAPON
Line 13 CAPABILITIES, NO PRESENT
Line 14 COMMITMENTS, SOME DEPLOYMENT NOT IN
Line 15 FRANCE AND CAPABLE OF
Line 16 REACHING THE UNIT'S
Line 17 DESIGNATED STAGING AREA WITHIN
Line 18 72 HOURS -- SHOW SEPARATE
Line 19 LISTINGS FOR EQUIPMENT
Line 2Ø LOCATED NEAR METZ AND TOULON.
Line 21 IF METRY EQ FR AND MECAP
Line 22 EQ D AND MESIC EQ Ø AND
Line 23 PLTRT LE Ø3ØØØØ.
Line 24 FURTHER IF MELOC CIR 49ØØNØØ611E/3Ø.
Line 25 SORT/1 LOC, PLTRT.
Line 26 FURTHER IF MELOC CIR 43ØØNØØ556E/5Ø.
Line 27 SORT/2 LOC, PLTRT.
```

Comment: An explanation of each of the preceding statements follows.

Line 1 -- The TITLE statement specifies that the name of this retrieval is SØ38R, the retrieval number is 11Ø3.

It will be processed by the RIT named TOPB38R; the run mode is compile-and-go.

Line 2 -- This statement specifies that the next two logical conditions are to be passed against the data file INSTALN.

Lines 3,4 -- This is a comment. It has no logic function.

Line 5 -- The POINT field is tested to see if it contains a coordinate within 3Ø miles of METZ.

Line 6 -- The SORT statement puts ØØ1 in the action code field of the sort key and the place name of the unit's location in the user-designated portion of the sort key.

Lines 7,8 -- The NOTE statement identifier designates these lines as comments.

Line 9 -- The IF statement tests for records having coordinates within 5Ø miles of TOULON.

Line 1Ø -- The SORT statement puts ØØ2 in the action code field of the sort key and the place name of the unit's location in the user-designated portion of the sort key.

Line 11 -- This statement causes the statements which follow to be passed against the TEST36Ø file.

Lines 12 through 2Ø -- Comments.

Lines 21 through 23 -- The IF statement tests for deployment country of France, dual weapons capability, no present commitments, and transportation times to the staging area less than or equal to 3 days.

Line 24 -- The FURTHER statement imposes an additional condition to the basic IF statement. It tests for all records having coordinates within 3Ø nautical miles of Metz.

Line 25 -- The SORT statement will cause the retrieved records to be sorted before the records retrieved by line 9 and line 26, because of the /1 following the word SORT.

Line 26 -- Another but separate FURTHER statement is imposed. The logic of this FURTHER is joined by an implied AND logical connector to test for all records having coordinates within 5Ø miles of Toulon.

Line 27 -- This SORT statement will cause the retrieved records to be sorted after the records retrieved by lines 5, 21-24, because of the /2 following the word SORT.

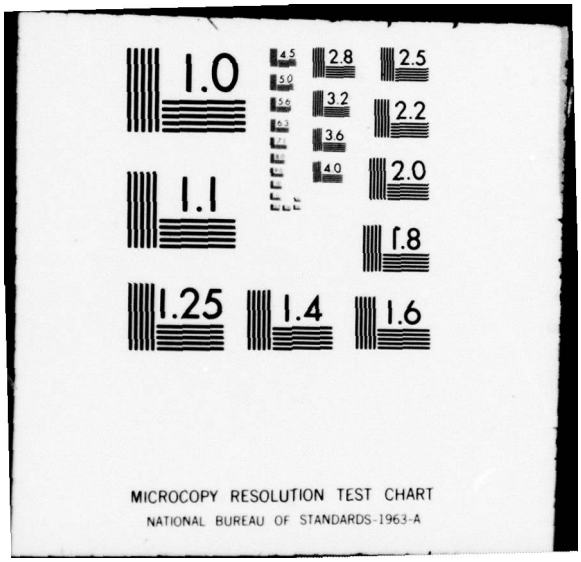
Following are the resultant entries in the Qualifying Record Table (QRT) generated for those data records which qualified. They are shown in their final sequence.

AD-A058 957 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 17/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78

UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-4 NL

2 OF 2
AD
A058957

END
DATE
FILMED
11-78
DDC


1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RETRIEVAL AND SORT PROCESSOR (RASP)

Retrieval Code	Type Code	Action Code	User Designated	Sort Key	Sequence Number	File Code	Cont. Code	Source
Ø9	Q	1	JOEUF		ØØ	A	Ø	INSTALN Record
Ø9	Q	1	JOEUF	Ø21123	ØØ	B	Ø	TEST36Ø Record
Ø9	Q	1	JOEUF	Ø21485	Ø1	B	Ø	TEST36Ø Record
Ø9	Q	1	METZ	Ø223ØØ	Ø2	B	Ø	TEST36Ø Record
Ø9	Q	1	ST MIHIEL		Ø2	A	Ø	INSTALN Record
Ø9	Q	1	ST MIHIEL	Ø111Ø8	Ø3	B	Ø	TEST36Ø Record
Ø9	Q	1	ST MIHIEL	Ø3ØØØØ	Ø4	B	Ø	TEST36Ø Record
Ø9	Q	2	TOULON		Ø3	A	Ø	INSTALN Record
Ø9	Q	2	TOULON	Ø11853	Ø5	B	1	TEST36Ø Record
Ø9	Q	2	TOULON	Ø11853	Ø6	B	1	TEST36Ø Record

Note: In merging data files, it is most important to ascertain that the common information employed for merge is of the same format and length.

Example 9

```
TITLE S039R/1203.

FILE TEST360.

NOTE -- ANY UNIT WITH MAJOR EQUIPMENT MODELS A-1,
     A-3, A-4, A-5, A-6, A-7, F-100, F-101
     F-105 TO BE DEPLOYED WITHIN THE NEXT 30 TO 60
     DAYS - SEQUENCE BY CLASS, TYPE AND DURATION.

IF MEMOD EQ A-1, A-3, A-4, A-5, A-6, A-7, F-100, F-101,
     F-105 AND MEDDT BT 68061/68120.

SORT MECLQ, MEDUR.
```

Comment: This illustrates a retrieval requiring a sort on information carried in the periodic set. If periodic sorting is used, only fields from one periodic set (one which was tested by the retrieval logic) may be specified in the SORT statement. When periodic sorting is utilized, only fields from those subsets which have been flagged will be put in the sort key. When all the terms of the IF statement have been processed against a data record, each subset is examined to determine if it has been flagged. When a flagged subset is found, a ROAER record is generated which points to the beginning location of the subset. The data from the field's major equipment class and type and the code for major equipment duration is placed in the sort key. RASP will continue to check the subsets for flags and output a new ROAER record each time a flagged subset is found.

Assume a data record from the TEST360 file contains four subsets in Periodic Set 1, having the following information in the fields illustrated.

Subset Seq. No.	MEDDT	MEDEP	MEDUR	MEMOD
ØØ1	68Ø72	F	R	A-5
ØØ2	68119	G	Q	A-1
ØØ3	ØØØØØ			SP-5A
ØØ4	68Ø37	G	S	F-1Ø5

Subsets ØØ1, ØØ2, and ØØ4 have satisfied the logic conditions and will be flagged. Three ROAER records will be generated and each will contain two Set Flags. A Set Flag tells the output processor the number of the set to which the information in the QDF belongs. It also points to the beginning of the set or subset placed in the Qualifying Data File, indicates any flagging, and indicates whether the QDF contains all or only selected subsets.

Separate records are generated for each flagged subset. Each of these records also contains the Fixed Set Flag and the user-designated sort key built, using information from the periodic subset.

If periodic sorting were not specified, the periodic set or subset Set Flags would be consolidated in the same record as a periodic group composed of three fields.

Example 10

```
SORT 'A',SERV,LOC 1/3 #D#, MEPSD,MEQPT #D#.
```

Comment: This illustrates a retrieval requiring several descending sort specifications. The result of the above specification would cause an ascending sort on the literal, 'A', and the fields, SERV and MEPSD, and a descending sort on the first three positions of the field LOC and the entire field MEQPT.

2.4.3.2 SELECT

The SELECT statement enables the user to designate the contents of an abbreviated answer record. It can be used only to select periodic sets and subsets and the variable set since the fixed set is included automatically. If only the fixed set is desired, a fixed set field must be the only operand.

The three options available may be specified in any combination in the same SELECT statement. If a set is referenced more than once in a SELECT statement or if the modes are mixed, the most inclusive output will be produced.

The format is

```
            field-name
SELECT      field-name/F     ...     .
            field-name/WF
```

where

SELECT (required) -- Is the statement identifier.

Field-name (required) -- Is the name of any field within the set selected to compose an abbreviated answer record. If used without a suffix, it selects all subsets in the set regardless of flagging.

/F (optional) -- Specifies that only flagged subsets of the referenced set be included.

/WF (optional) -- Specifies that the entire set referenced be included, but only if it contains one or more flagged subsets.

.(Required) -- Is the statement terminator.

Example 1

```
IF ATTACH EQ NATO AND UNTYZ NE 2 AND MECL EQ A$, M$ AND

MEDEP EQ Ø AND MEMOD NE EC-121, TC-121.

SELECT MEQPT/F, PLAN.
```

Comment: MEQPT is in Periodic Set 1 and PLAN is in Periodic Set 3. The QRT will contain entries pointing to the fixed set, only those subsets from Periodic Set 1 flagged, and all of Periodic Set 3. A SORT statement is implied immediately preceding the SELECT statement, even though no sort fields are designated.

Example 2

```
IF SERV EQ W AND UNTYP EQ

22KBA, 22JCA, 22FBA, 22ECA,

22XCA, 22KHA

AND MEMOD EQ CGM-13, MGM-13, XMGM-31.

SORT UNLVL, UNTYP, CNTRY.

SELECT MEQPT/F.
```

Comment: SERV and UNTYP are in the fixed set and MEMOD is in Periodic Set 1. The QRT will contain entries pointing to the fixed set and the subsets flagged in Periodic Set 1.

Example 3

```
IF MESIC NE Ø.

SORT UIC, LOC.

SELECT MESIC/WF.
```

Comment: This retrieval will present all periodic subsets in Periodic Set 1 if the unit has any equipment on special alert status. If the /F modifier were used, only those subsets flagged would be selected. In either case, the fixed set containing unit identification information will be included.

2.5 Merged File Retrieval

All of the rules which are applicable to single file retrieval can be applied to merged file retrieval which is the NIPS capability to interrogate up to 10 files in a single job step. The RASP cataloged procedure has been written to handle up to three files; any additional files must have DD statements temporarily added to the procedure to identify them.

Merged file retrieval allows the user to query more than one file and merge the retrieved data records through a common sort. Choice of the fields to be sorted on for each file is immaterial; however, the records are merged together according to the fields chosen for sort in the retrievals from the individual files. Generally, the fields are similar type fields. Logically, in merging data files, it is most important to ascertain that the common information employed for merge is of the same format and length.

In merged file retrieval a control division is still necessary; that is, the entire retrieval must begin with a FILE statement. The file to be first queried must be referenced here. The control division may also contain the RASP library action statements: EXECUTE and/or DELETE. No primary LIMIT may be used.

Following the control division of the query must be a TITLE statement identifying the single retrieval that may be included. Here the retrieval must be named and assigned a retrieval number. Other optional TITLE statement operators may be included as appropriate.

Following the TITLE statement the first file to be queried may be referenced again in a FILE statement; however, if omitted the first file to be queried will

default to the file identified in the control division FILE statement. The limit/retrieval/sort/select statements that apply to the first file queried follow as applicable.

The second file to be queried is identified with another FILE statement which is then followed by the limit/retrieval/sort/select statements for that file.

Any subsequent files are identified and queried in the same manner.

Example 1

```
FILE INSTALN.
TITLE S038R/1103.
FILE INSTALN.
IF POINT CIR 4909N00611E/30.
SORT '1' LOC.
IF POINT CIR 4300N00556E/50.
SORT '2' LOC.
FILE TEST360.
IF METRY EQ FR
   AND MECAP EQ D.
FURTHER IF MELOC CIR 4900N00611E/30.
SORT '1' LOC PLTRT.
FURTHER IF MELOC CIR 4300N00556E/50.
SORT '2' LOC PLTRT.
```

Comment: This example illustrates a merged file retrieval executed against two files: INSTALN and TEST 360. The first file to be queried is INSTALN; therefore, it is identified in the control division FILE statement. The TITLE statement assigns the retrieval name S038R, and a retrieval number of 1103. The next statement is a repeat of the control division FILE statement and again identifies INSTALN as the first file to be queried. This statement may be omitted and will default to the file on the control division FILE statement if it is. In either case INSTALN is to be interrogated by the RASP statements in the retrieval until the FILE TEST360. statement is encountered.

In the INSTALN retrieval, records concerned with military airfields within 30 nautical miles of METZ and 50

nautical miles of TOULON are retrieved and sorted on location (LOC). A SORT literal of 1 is inserted into the SORTKEY for those records with locations near METZ and a 2 for those records with locations near TOULON.

When the end-of-file of the INSTALN file is reached, the second file, TEST360, then becomes the file being queried. Here records concerned with deployments in France with dual weapon capabilities, and located near METZ and TOULON are retrieved. Again, a SORT literal of 1 is inserted into the SORTKEY for a record with a location near METZ anda 2 for those near TOULON.

The retrieved records will be merged together in the order specified by the SORTs. All the records from both files identified by the SORTKEYs containing 1 will appear first. The records from both files identified by the SORTKEYs containing 2 will appear last. Merged file output is illustrated in Users Manual, Volume V, Output Processor(OP), paragraph 5.1.3.

2.6 Restrictions

The following maximum limits are imposed by the RASP component on each job (batch of retrievals).

a. Maximum number of SORT or SELECT pairs per IF or FURTHER statement - 20.

b. Maximum number of RIT-names per retrieval = 50.

c. Maximum number of statements per retrieval = 256.

d. Maximum number of subroutines per retrieval = 20.

e. Maximum number of clauses per statement = 256.

f. Maximum number of retrievals in a single file job = 96.

g. Maximum number of retrievals in a merged file job = 1.

h. Maximum number of files in a merged file job = 10.

i. Maximum number of field or group names per retrieval = 400.

j. Maximum number of replaceable variables per retrieval = 255.

k. Maximum number of cards per statement = 27.

l. Maximum number of characters comprising a list of multiple data values in a clause: N = 250 where N is total of all characters in the list plus three for each value.

m. Maximum number of operands for all SORT statements in one RASP job is 340.

2.7 Defining a Skeleton Query

A skeleton query is defined by prefixing at least one operand in the query with an underscore character (0-5-8 punch). The presence of this character automatically causes that query to be generated so that the operands so designated can be replaced by other operands for succeeding executions. The operands cannot be replaced during the same run in which the query is being compiled.

Operands which are legal for designating as replaceable are the value portions of condition clauses, Function Operator names, members of the Function Operator parameter list and SORT statement terms.

Some basic rules for designating replaceable operands (variables) are:

a. Replace a literal with a literal.

b. Replace a field name with a field name.

c. Replace field names with field names from the same set.

There are other rules or exceptions to rules which apply to particular statements or clauses. These are defined in succeeding paragraphs.

2.7.1 Standard Condition Clause Variables

The value portion of a condition can be designated as replaceable.

Example 1

```
....MEQPT EQ _TANK....
```

Comment: TANK is designated as a replaceable item; thus, at execution time it can be replaced. If it is not replaced, data field MEQPT will be compared to the literal TANK.

The same value might be designated as replaceable a number of times. In this case, a single replacement will cause all the like values to be replaced.

Example 2

```
...MERDY  LT _10 OR MEPSD GT _10 OR MEPEQ EQ _10...
```

Comment: Numeric 1Ø is designated as replaceable in three different condition clauses; thus, the replacement of 1Ø would cause all three clauses to use the replacement value.

Subroutine conversion of a replacement literal value cannot be specified at execution time. However, if the replaceable variable was to be converted, the replacement variable will be converted. This is true only for condition clauses (see restriction for sort variables in subsection 2.6.3). A replacement variable replacing more than one replaceable variable may be converted in one case and not in the other, or even converted by a different conversion subroutine, depending upon the original replaceable variable specification. In either case, at qualification time, the appropriate value will be used to qualify the data.

Example 3

```
...CNTRY EQ _SLOBOVIA OR LOC EQ _SLOBOVIA...
```

Comment: Assume that CNTRY has a conversion subroutine designated in the FFT and that LOC does not. At execution time, SLOBOVIA is replaced by ROSSLYN. ROSSLYN would be converted by the subroutine, and the converted value would be used in the first condition. The unconverted value would be used for the second condition.

A data field name (indirect addressing) can be replaced. However, the replacement must also be a data field name and must belong to the same set as the original data field name. Note that the data types must be compatible; i.e., alpha fields are compared to alpha fields, numeric to numeric, and coordinate to coordinate.

Example 4

```
...MEPSD GT __&MEREQ...
```

Comment: Data field name MEREQ can be replaced by another data field from the same set as MEREQ.

A condition can be omitted at execution time if a value has been designated as replaceable and is replaced by a dollar sign. A single dollar sign ($) is the only valid usage of a dollar sign as a replaceable value. Care should be used with this option and particularly with periodic conditions. If a condition is omitted and that condition is part of an AND string, the omitted condition is treated as a true condition. Therefore, if all conditions against a periodic set are omitted and the record qualifies, subsets for that set would be flagged.

Example 5

```
FILE TEST36Ø.
TITLE RET1/Ø1  NO-GO.
IF CNTRY EQ USA AND MEQPT EQ _TANK
AND PLEAC EQ A.
```

Comment: CNTRY is a field in the fixed set, MEQPT is a field in Periodic Set 1, and PLEAC is a field in Periodic Set 5. The value TANK has been designated as a replaceable variable.

The following EXECUTE statement would cause all records which qualify on CNTRY and PLEAC to qualify regardless of MEQPT.

Example 6

```
EXECUTE RET1(TANK=$).
```

Comment: In this example, all subsets of Periodic Set 1 would be flagged.

An omitted condition which is in an OR string will be treated as false. Thus, flagging of subsets would logically be correct.

Example 7

```
FILE TEST36Ø.
TITLE RET1/Ø1  NO-GO.
IF CNTRY EQ USA OR MEQPT EQ _TANK
OR PLEAC EQ A.
```

The following EXECUTE statement would cause qualification of the record if, and only if, CNTRY or PLEAC is true.

Example 8

```
EXECUTE RET1(TANK=$).
```

Comment: Subsets contained in Periodic Set 1 would not be flagged.

Literal replacement values for a conditional clause may be a multiple value replacement; i.e., a single literal value in a conditional clause may be replaced by several values. These values will be restricted to use the EQ, NE,

CONTAINS, and BT relational operators. These values must be enclosed in parentheses.

Example 9

```
FILE TEST36Ø.
     TITLE RET1/Ø1  NO-GO.
     IF MEQPT EQ _TANK.
```

Comment: TANK could be replaced by the multiple values of SHIP, PLANE, and GUN, by use of the following EXECUTE statement:

```
EXECUTE RET1 (TANK=(SHIP,PLANE,GUN)).
```

At execution, the conditional statement thus becomes:

```
IF MEQPT EQ SHIP OR MEQPT EQ PLANE OR MEQPT EQ GUN.
```

There are three relational operators in which the clause format is somewhat special. These are the BETWEEN (BT), the Overlapping Polygon (OVP), and Circle Search (CIR). Replacement designation for each is discussed in succeeding paragraphs.

Either or both of the value for the BT operator can be designated as replaceable.

Example 10

```
IF MEPSD BT  Ø/_1Ø.
IF MEPSD BT _Ø/1ØØ.
IF MEPSD BT _Ø/_1ØØ.
```

Comment: In the first example, the value 1Ø could be replaced by another value; in the second example, the value 0 could be replaced by another value; in the third example, either or both values could be replaced.

If either of the factors is replaced with the $ character, the condition is omitted. At the time the retrieval is compiled, a check will be made to see if a logical relationship exists between the two operands. If

the first operand is greater than the second, the values will be reversed and stored in that manner.

For example:

```
IF MEPSD BT_10/_01.
```

will be stored as if it reads

```
IF MEPSD BT_01/10.
```

and

```
IF MEPSD BT_START/END.
```

will be stored as if it reads

```
IF MEPSD BT_END/_START.
```

This second case would probably not be what the user intended, so care should be exercised to use replaceable value names which are in the proper collating sequence to achieve desired results. No check for a logical relationship is made after replacement and during execution of the query. Therefore the effective result could be BT 200/100.

If either of the factors is replaced with the $ character, the condition is omitted.

Any or all designated coordinates for the OVP operator can be designated as replaceable. Also, a coordinate can be replaced by more than one coordinate. The replacement coordinate(s) are merged with those which are not designated as replaceable or which are not replaced. If there are no coordinates after the merge has taken place, the OVP is treated as an omitted condition.

Example 11

```
IF MEPNT OVP 4900N0065ØE/4940N0615E/_4945N0220E.
IF MEPNT OVP _ØØØØNØØØØØE.
```

Comment: In the first example, the third parameter could be replaced by from one to six coordinates (OVP allows a maximum of eight coordinates). These would be merged with the first two parameters to form the polygon. The second example illustrates a more practical use of the replacement variable with the OVP operator. This is where the user defines a dummy polygon consisting of a single coordinate and at execution, defines his true coordinate(s).

Either or both the coordinate and/or the distance can be designated as replaceable for the CIR operator.

Example 12

```
IF POINT CIR 494ØN ØØ615E/_35Ø.
IF POINT CIR _494ØN ØØ615E/35Ø.
IF POINT CIR _494ØN ØØ615E/_35Ø.
```

Comment: In the first example, the value 35Ø could be replaced at execution time. In the second example, the value 494ØN ØØ615E could be replaced. In the third example, either or both values could be replaced. If either value is replaced by a $ character, the CIR condition would be omitted.

2.7.2 FUNCTION Operator Replaceable Variables

Both the FUNCTION Operator subroutine name and/or parameters in the subroutine parameter list can be designated as replaceable variables and can be replaced at execution time.

To designate a FUNCTION Operator subroutine as replaceable the underscore character must be prefixed to the subroutine name. Care must be taken that the replacement subroutine and the original parameter lists are compatible.

Example 1

```
...AND FUNCTION _ADD &MEORC,&MEORN,WORK1 AND...
```

Comment: FUNCTION Operator ADD can be replaced. However, the replacement operator should expect two values and a system work area as its parameter list.

Parameters in the FUNCTION Operator subroutine parameter list are designated as being replaceable by prefixing that parameter with an underscore character. Data fields which are replaced are checked to determine whether they are replaced by another data field from the same set as the original specification. Literals must be replaced by literals. A literal parameter can only be replaced by a single value literal. A literal does not have to be replaced by a literal of the same mode; i.e., an alpha literal might be replaced by a numeric literal. If the replacement literal is all numeric characters it will be converted and passed to the FUNCTION subroutine as a full word binary value.

Example 2

```
    ...AND FUNCTION GCD _&POINT _6Ø75N12535E,
_9ØØ GT WORK1 OR...
```

Comment: Data field POINT could be replaced by another coordinate data field from the same set. Latitude/longitude 6075N12535E could be replaced by a value which would be given to the FUNCTION Operator as an EBCDIC string of characters (includes alpha characters). 9ØØ could be replaced by a numeric value.

2.7.3 Sort Statement Variables

Fields which compose the sort key may be designated as replaceable. This is accomplished by prefixing the sort field designation with an underscore character. A literal can only be replaced by a literal, and a data field can only be replaced by a data field from the same set. The descending sort flag, #D#, may not be specified as replaceable. However, if the field or group name preceding the descending sort flag is marked as replaceable and the field or group is replaced, then the descending sort option will not be taken. See examples 4 and 5 for an illustration of replacing fields with descending sort specified.

Example 1

```
SORT CNTRY, _MEQPT, _'X'.
```

Comment: The field MEQPT can be replaced by another field from the same set as MEQPT. The literal X can be replaced by another literal. If the Output Processor references relative positions within the sort key as specified in the user's RIT, the relative positions may change unless the replacement value is of the same length.

If an operand in the SORT statement is replaced by a $ character, that data field or literal will be omitted from the sort key.

Subroutine conversion of a data field cannot be designated for replaceable fields. The sort replacement differs from the condition replacement in this respect. Regardless of whether subroutine conversion was performed on the original operand, no conversion will be performed with the replacement field.

Example 2

```
FILE TEST36Ø.
TITLE RET1/Ø1 NO-GO.
IF MRPSD GT 0.
SORT _ CNTRY #CNTRYS#.
```

Comment: At execution time, the field CNTRY could be replaced as follows:

```
EXECUTE RET1 (CNTRY=UNTOE).
```

The results of the query would be sorted by the field UNTOE rather than by CNTRY. Also, the subroutine conversion would not apply to UNTOE.

In the above example, if the user wished to sort by CNTRY unconverted, it could be done with the following EXECUTE statement:

```
    EXECUTE RET1 (CNTRY=CNTRY).
```

Partial-field notation and subroutine conversion are processed similarly; i.e., PFN cannot be designated for the replacement value, and the skeleton notation does not apply to the replacement.

Example 3

```
    FILE TEST36Ø.
    TITLE RET1/Ø1  NO-GO.
    IF MEPSD GT Ø.
    SORT _TRDTG 1/6.
```

Comment: At execution time, the field TRDTG could be replaced as fcllows:

```
    EXECUTE RET1 (TRDTG=RADTG).
```

The partial-field notation would not apply to the field RADTG, and the results of the retrieval would be sorted on the entire contents of the field RADTG.

Example 4

```
    FILE TEST36Ø.
    TITLE RET1/Ø1 NO-GO.
    IF UIC GT MØØØØ1.
    SORT UIC _PERS #D#.
```

Comment: At execution time, the field PERS could be replaced as follows:

```
    EXECUTE RET1 (PERS=PERS)
```

The descending sort option would not be applied to the field PERS, and the results of the retrieval would be an ascending sort on both fields, UIC and PERS.

Example 5

```
FILE TEST360.
TITLE RET1/01 NO-GO.
IF UIC GT M00001.
SORT UIC _LOC #D#.
```

Comment: At execution time, the field LOC could be replaced as follows:

```
EXECUTE RET1 (LOC=CNAM).
```

The descending sort option would not be applied to the field CNAM, and the results of the retrieval would be an ascending sort on both fields, UIC and CNAM.

2.8 Writing a FUNCTION Operator Subroutine

This subsection is primarily directed towards the programmer responsible for writing the subroutine. The terminology used is programmer-oriented. Based on the qualification algorithm established by the user (analyst) and based on any additional output requirements, the programmer can proceed with detailed design and coding of the subroutine. The subroutine should be written as a single root segment, and no input or output functions should be performed by the user routine.

The Function Operator subroutine should be compiled and link edited onto the Partitioned Data Set Library, which is used to store program load modules and is included in the SLIB concatenation of data sets when executing RASP. (This library is frequently referred to as the User File Library or Composite Library and contains retrievals, RITs, conversion tables and subroutines.)

The following paragraphs describe the interface or linkage required for Function Operator subroutines. RASP uses the standard NIPS 360 FPS Subroutine Supervisor (SUBSUP) calling sequence. Input formats can usually be predetermined; thus, the use of COBOL and/or FORTRAN is feasible. More complex and variable input subroutines will require use of OS/360 Assembler Language. The RASP Function

Operator capability provides the interface with the user routine to cause execution of the qualification algorithm and to pass generated data to the QRT for display or further processing by the Output Processor.

Passed Data String Format

RASP will "gather" data elements of the Function Operator parameter list as prescribed by the user in his query and will formulate a data string which will be passed to the named subroutine. In this string, there may be elements from the data file, from literals within the parameter list or from the system work areas. All numeric data from the data file and system work areas, and from unsigned numeric literals, will be passed as fullword binary values.

Signed numeric literals are passed as EBCDIC literals and must be converted to the proper form by the subroutine. The maximum or minimum value passed as a fullword binary value is ±2147483647. The passed data string format is as follows:

AABBCCDEFF..FFCCDEFF..FF....

where

A = Number of bytes in the data string

B = Number of data elements in the data string

C = Number of bytes (length) of the data element

D = Type of data element code

A = Alpha (decimal)

B = Binary

C = Coordinate

F = Set code (Ø = fixed set, 1 = periodic set, 2 = work area, 3 = literal)

F = Data value.

C, D, E, and F are repeated for each item in the parameter list. If a parameter is replaced by use of the replacement capability, the length and mode for the replaced value will be used. Obviously, if the replacement length is different from the original length, format problems will be encountered when predefined fixed formats are used. Note also that a $ mask replacement value will result in a data element length of zero, and that mode and set code will not be set.

Alignment will not be guaranteed; however, if all numeric values appear first in the parameter list, they will be fullword aligned.

Calling Sequence to the Function Operator Subroutine

The standard SUBSUP (NIPS 36Ø FFS Subroutine Supervisor) calling sequence to user subroutines will be used. This conforms to the standard OS/36Ø linkage conventions. Three parameters are passed to the subroutine. They are the Entry Point to SUBSUP (for subroutines which call other subroutines), the data string and a result byte. The Entry Point to SUBSUP can only be used by Assembler Language subroutines. However, a "dummy" definition in COBOL and FORTRAN subroutines must be defined. The data string will be used by all subroutines regardless of compiler language. The result byte is for placement of a code which will be used in the retrieval qualification. In Assembly Language routines, Register 15 may contain the return code. The codes are S if true and M if false. It must be set upon return from the subroutine.

A calling sequence example is as follows:

```
CALL SUB (P1, P2, P3) or
CALL 'SUB' USING P1, P2, P3.
```

Where

SUB = Name of Function Operator subroutine

P1 = SUBSUP Entry Point

P2 = Data string (described above)

P3 = Result byte.

Assembler Language Subroutine Linkage

Standard OS/360 linkage conventions should be used, and the routine should use the following macro as its first instruction.

```
SUBNAME FFSBEGIN BASEREG
```

This macro will generate the proper CSECT and SAVE linkage. X13 will point to a generated SAVE area and should not be used by the qualification routine. Register BASEREG will have been initialized as the routine base register along with the appropriate using statement. Register 1 will point to the parameter address constant list. When returning control, Register 15 may contain the return code as discussed previously; the following macro will be used to return control:

```
FFSRETRN RC=(15)
```

Otherwise, the byte indicated by parameter three must be filled with 'S' or 'M'.

COBOL Subroutine Linkage

COBOL Function Operator subroutines require a LINKAGE SECTION and an ENTRY. The 'linkage section' describes the parameters which are passed to it. The first parameter (SUBSUP Entry Point) must be defined, even though the subroutine cannot use it. The second parameter describes the data string for this particular function subroutine and the third parameter is the result byte.

The following is an example of the LINKAGE SECTION:

```
01   P1.
     02   NOTHING PICTURE X.
01   P2.
     02   DSLEN PICTURE S(99) USAGE COMPUTATIONAL.
     02   DSNUM PICTURE S(99) USAGE COMPUTATIONAL.
     02   DSP1LEN PICTURE S(99) USAGE COMPUTATIONAL.
     02   DSP1TYP PICTURE X.
     02   DSP1SET PICTURE X.
     02   DSP1VAL PICTURE XXXXX.
     .
     .
     .
01   P3.
     02   RETURN-CODE PICTURE X.
```

Entries DSP1LEN through DSP1VAL would be repeated to describe each data element which was input to the routine (with different field names). Of course, the description of the value itself would vary.

The ENTRY statement will be located somewhere in the PROCEDURE DIVISION of the function subroutine. The following is an example of the ENTRY statement:

```
ENTRY 'SUB' USING P1 P2 P3.
```

FORTRAN Subroutine Linkage

If all input values to the FORTRAN subroutine are assumed to be numeric, the data string can easily be defined. The TYPE and SET bytes must be ignored. The FORTRAN subroutine must contain a SUBROUTINE statement and statements declaring or defining the data.

The following is a FORTRAN example:

```
SUBROUTINE      SUB (P1,P2,P3)

DIMENSION       P4(5),P5(10),P2(3)

INTEGER*2       P5
```

```
LOGICAL*1       P3

EQUIVALENCE     (P4(1),P5(1))

DATA            S/'S'/,M/'M'/
```

In the preceding example, assuming that two data elements are being passed to the subroutine, the data string is moved into the subroutine and into an area which is defined as five fullwords. The same area is defined (equated) to 10 halfwords, thus, allowing the program to work with either halfword or fullword values.

The result code is defined as one byte. A value which results in the appropriate code (S or M) must be placed in the result prior to returning. Also, the HOP of the data string will be fullword aligned; thus, all alignment will fall in place, provided that only numeric parameters are specified.

2.9 Writing an Analyzer Subroutine

This section is intended for the programmer responsible for writing the subroutine. All conditions, rules, and restrictions regarding a FUNCTION Operator as described in sections 2.4.2.2.3, FUNCTION Operator, and 2.8, Writing a FUNCTION Operator Subroutine, apply to an analyzer subroutine. Therefore, this section will only explain the passed data string format for an analyzer subroutine. The calling sequence to the analyzer subroutine and subroutine linkage using Assembler Language, COBOL, or FORTRAN are exactly as described in section 2.8, Writing a FUNCTION Operator Subroutine.

Index Processing will create a data string from the data elements of the FUNCTION Operator parameter list. In this string, there may be user name fields or literals. There may be no elements from the data file, from the Index Data Set, or from any system work area. All literals will be passed exactly as they are submitted on the input statement but without the delimiting quote marks. No conversion of any kind will be performed.

The passed data string format input to the analyzer routine is as follows:

```
AABBCCDEF.....FCCDEF....F...

     Data        Data
     Element 1   Element 2
```

where:

A - Number of bytes in the entire data string (binary)

B - Number of data elements in the string (binary)

C - Number of bytes in the data element (binary)

D - Type of data element code - One byte (alpha)

For the indexed field, this is the data file mode; otherwise:

A - Alpha (a literal)

F - A field name (nonindexed field or not the first indexed field)

E - Status code - One byte (alpha)

0 - Indexed field in the clause

1 - Nonindexed field

3 - Literal

F - Data value

C, D, E, and F are repeated for each item in the parameter list. If a parameter is replaced by use of the replacement capability, the length and status of the replacing value will be used. No alignment is guaranteed.

As noted above, the calling sequence for an analyzer subroutine is the same as for a FUNCTION Operator. An example of the calling sequence is:

```
CALL 'ANLYZ' USING P1 P2 P3.
```

where:

ANLYZ - Name of analyzer subroutine

P1 - SUBSUP entry point

P2 - Data string as described above

P3 - Result byte

S - Indexes are to be used

M - Indexes are not to be used

The subroutine will set the result byte based on its analysis. If indexes are to be used, the subroutine must return data in the same area as the passed data string (parameter P2, described above). The size of the data string returned cannot exceed 256 bytes.

Format of the data string output from the analyzer subroutine is as follows:

```
AABBCD......DCD......D.......

  Data     Data      Data
  Element  Element   Element
```

where:

A - Number of bytes in the entire data string as returned by the subroutine (binary).

Note: This string cannot exceed 256 bytes.

B - Number of data elements returned by the subroutine (binary)

C - Number of bytes in the data element (one byte - binary)

D - Data elements

The values returned will be used as search arguments against the Index Data Set, as if the clause were field EQUAL value returned by the analyzer routine.

2.10 SYSDATS FUNCTION Subroutine

SYSDATS is a FUNCTION subroutine which allows the user to access the current system date for use in retrieval. By using the subroutine, records can be retrieved based on the logical relation between a file date field and the current system date. The relationship is established by applying an adjustment factor to the system date and then testing this adjusted date against the file date field for the specified relationship.

2.10.1 SYSDATS Format

The format for using SYSDATS is:

```
FUNCTION SYSDATS relop &fld 'adj1' 'adj2'
```

where:

FUNCTION (required) -- used to identify the string as FUNCTION items.

SYSDATS (required) -- used to identify the FUNCTION subroutine being invoked.

Relop (required) -- the relational operator specifying the type of test to be performed. The valid operators are EQ, NE, LT, LE, GT, GE, BT, and NB. Each of these operators is a standard RASP relational operator except for NB, which represents NOT BT.

&fld (required) -- the file fieldname which contains the date value. The field must be a 6-character

ALPHA/DECIMAL field in the form of YYMMDD. The field name must be preceeded by an ampersand (&).

'adj1' (required) -- the adjustment factor to be applied to the system date. The format of the adjustment is:

'snnYRnnMOnnDA'

where:

s (optional) - a sign (+ or -) indicating that the adjustment is to be upward (+) or downward (-). When omitted, the adjustment is assumed to be upward.

nn (required) - a 1 to 2 digit number specifying the number of years, months and/or days the system date is to be adjusted. The number can be any number from zero (0) to 99.

YR - indicates an adjustment in years.

MO - indicates an adjustment in months.

DA - indicates an adjustment in days.

The adjustment factor must be enclosed in single quotes. The adjustment may be specified using one or more of the time units, for example:

'-7DA' - is seven days prior to the current system date.

'+3MO7DA' - is three months and seven days after the current system date.

'1YR1DA' - is one year and one day after the current system date, upward adjustment assumed.

If the current system date is to be used without adjustment, the user would code the adjustment as zero (0).

The adjustment factor as applied must not cause the system date to overlap the century.

'adj2' (optional) -- is a second adjustment factor to be applied to the current system date only when either the BT or the NB relational operator is being used. The format 'adj2' is the same as the format of 'adj1'. When used, the resultant adjusted date must be greater than or equal to the adjusted date derived from 'adj1'.

2.10.2 SYSDATS Examples

```
IF FUNCTION SYSDATS EQ &FDATE '0'.
```

This example will retrieve records where FDATE is equal to the current system date.

```
OR FUNCTION SYSDATS GT &FDATE '-7DA'.
```

This example will retrieve records where FDATE is greater than seven days prior to the current system date.

```
AND FUNCTION SYSDATS LE &FDATE '6MO'.
```

This example will retrieve records where FDATE is less than or equal to six months after the current system date.

```
FURTHER FUNCTION SYSDATS BT &FDATE '-6DA' '0'.
```

This example will retrieve records where FDATE ranges from six days prior to and up to the current system date.

Section 3

INPUT

RASP will retrieve from data files organized sequentially on magnetic tape as single data files or as concatenated segments of a data file, or as either an indexed sequential data set or a virtual storage access data set resident on a Direct Access Storage Device (DASD).

3.1 Data Files

RASP will retrieve from data files organized sequentially on magnetic tape or as either an indexed sequential or a virtual storage access data set resident on a Direct Access Storage Device (DASD). The data set name for the data files may be a qualified data set name up to 44 characters in length. When retrieving from a tape data file, the Queued Sequential Access Method (QSAM) will be used as will the Queued Index Sequential Access Method (QISAM) when retrieving from an ISAM data file or the Virtual Storage Access Method (VSAM) from a VSAM data file residing on a Direct Access Storage Device. The FPT, which describes the logical record structure of the data file, is stored as a part of the data file.

3.2 Index Data Set

This data set is the repository of all indexing information for its associated data file. The data set name is the data file's qualified or unqualified name, less the "S" for SAM files, with the suffix "X."

3.3 Procedure Library

The Procedure Library contains, among others, a set of Job Control statements for RASP that have been placed in a special data set (SYS1.PROCLIB) and that can be retrieved by

naming it in the OS/360 execute (EXEC) statement. The procedure, XRASP, contains the job and program information, data characteristics and device requirements used by the operating sytem to regulate the execution of RASP job steps, allocate input/output resources, obtain and dispose of data, and communicate with the operator.

3.4 Program Library

The Program Library is a partitioned data set that contains user-written subroutines and tables, retrievals in a program format suitable for loading into main storage for execution, the RASP programs themselves, and standard system-supplied conversion subroutines and tables. The library data set name may be a qualified data set name up to 44 characters in length.

3.5 RASP Control and Source Statements

The RASP Control and Source statements define the environment for the RASP job, provide the logical conditions for one or more retrievals, and specify the library maintenance action to be taken against the Permanent Retrieval Library, a subset of the Program Library.

3.6 Overriding Index Processing

Index Processing normally activates the candidate-access mode if at least 75% of all retrievals in one run can utilize index information. The user can override this by coding an entry on the EXEC statement. The entry is PARM='INDEX=XXXX', where XXXX is one of the following:

- NO - Override Index Processing entirely. RASP will not activate Index Processing at all.
- EVERY - Allow the candidate-access mode only if all retrievals in a batch can utilize index information.
- ANY - Allow the candidate-access mode if any retrieval in the batch can utilize index

information, regardless of the number of universal queries.

However, the presence of a KEYWORD statement in the retrieval negates the PARM override so that indexing is always invoked when the KEYWORD statement is found.

Section 4

OUTPUT

This section provides a general definition of the various forms of output.

4.1 Qualifying Data File (QDF)

The Qualifying Data File (QDF) is written, using the Basic Sequential Access (BSAM), as a sequential file of variable length blocked records on a Direct Access Storage Device in Record ID sequence. Each logical block which satisfies the conditions of one or more retrievals is written on the file. A data record is written only once. The QDF is not sorted and contains no sort keys.

The QDF also includes records which contain file classification, file control data, the FMS logic statements, and source-form retrieval statements for each file.

4.2 Qualifying Record Table (QRT)

The QRT is a sequential file consisting of variable-length blocked records. One of the primary records in this table consists of a sort key and a pointer to the fixed field of a data record on the Qualifying Data File which satisfied the conditions of a retrieval. There is at least one such entry for each record that qualified. Also included in each QRT entry will be information identifying the subsets in each data record which caused the retrieval to be satisfied. There are also records identifying the files used and records identifying the RIT names used.

4.3 Transaction Confirmation

The source-form of all retrievals in a job is listed on the on-line printer to provide a permanent record of the retrievals submitted in each batch. Errors detected during the edit pass are flagged and error codes are printed in the left margin opposite the invalid statement.

4.4 File Indexing Statistics and Messages

Errors detected by the file indexing function are communicated to the user by an error code and text as appropriate. As indexing operates on retrievals that are essentially error-free, the retrieval statements will not be printed out. The clauses that activate index usage (i.e., the must-satisfy clauses referencing indexed fields) are printed.

Index Processing prints the number of candidate records that will be examined by Retrieval Proper to determine the qualifying records.

4.5 File Analysis and Run Optimization Statistics

The File Analysis Statistics Capability in the RASP component provides transactions for each query executed in a RASP execution. The data set name (DSNAME) of this data set must be the data file name suffixed by a T. The T is added to ISAM and VSAM names; the S is replaced by T in SAM names. To obtain transaction output, the DSNAME must be cataloged and the user must specify the volume serial (VTRANS) and unit (UTRANS) in the execution procedure. The volume may be any direct access volume.

If the transaction data set exists at execution time, transactions will be added (DISP=MOD). If it does not exist, a five track data set will be dynamically allocated. The user may change the allocation value by overriding the TRANST DD card space parameter. Transactions are written as fixed length, unblocked, 50-byte records. The format (fixed) and length (50) cannot be changed but the user may change the blocking factor by specifying a DCB BLKSIZE in the TRANST DD card which is a multiple of 50.

If the user specifies a DSNAME (TRANS) in the TRANS DD card, he must supply all parameters required to process the data set. These parameters must conform to the requirements defined above

The Run Optimization Statistics capability provides the user with statistical data reflecting the core allocation during a RASP execution. The breakdown of the statistics details the amount of core used for user subroutines and tables, queries, stash area, I/O buffers, and access methods. It also includes the number of BLDL entries allocated and used and the number of entries required for all subroutines, tables, and the query to reside in core. The amount of core required for each subroutine and table and the query to reside in core will also be output. If subroutines and tables are rolled, this information will be output with the causes for the rolling and the number of times it occurred.

The user is able to enter override parameters for the number of BLDL entries to allocate, and the size of the stash area to be used for storage of data records.

The statistics gathering is initiated through parameters entered in the PARM field of the EXEC card. The parameters and their functions are as follows:

ROS - Gather and output Run Optimization Statistics.

NOROS - Omit statistics. If no other parameter is used, this parameter should be omitted, as it is the default.

The parameters the user may supply to tailor core allocation are as follows:

TCP=nK - The number (n) of 1000 (K) bytes to allocate to stash area.

TCB=n - The number (n) of BLDL entries to allocate.

TCS - Invalid in RASP.

With the above override parameters, run optimization statistics will be gathered and output unless NOROS is also entered.

For a more detailed description of the capability, see Volume I, Introduction to File Concepts.

4.6 Operator Messages

The RASP component is designed to operate with minimum operator intervention. All operator-system communication is performed under control of the Operating System.

Section 5

JOB MANAGEMENT

CSM UM 15-74, Volume VIII, Job Preparation Manual, illustrates the use of the cataloged procedure XRASP. It may be used for single file retrievals or for merged-file retrievals. The "symbolic parameter" capability provided by the OS/360 is utilized by the procedure. This provides the user with a simple method of designating the data sets to be used for a given run.

A standard set of defaults have been specified for the symbolic parameters. Assuming that all data file and library data sets have been cataloged for the typical run, only the data file and library require naming at run time. Default parameters, a listing of the XRASP procedure and DD name usage are provided in the Job Preparation Manual.

The procedure has been defined to suffix the names of sequential data files with "S," Index Data Sets with "X," and user libraries with "L." This convention is standard for all components in NIPS 360 PPS. These suffix characters must be taken into consideration when cataloging the data sets and when using them at execution time.

5.1 Checkpoint/Restart

During SAM processing, the user may invoke the OS/360 checkpoint/restart capability to record timed or end-of-volume checkpoints. If checkpoints are needed during ISAM processing, only the timed option is valid. The checkpoint/restart capability should only be used during long-running jobs using the execute only procedures. (Note that the OS/360 step restart is program independent and is not the topic of this discussion.) A detailed description of the OS/360 check/restart capability (which is utilized in

NIPS) is available to the interested user in IBM Systems Reference Library, Number C28-6708. A detailed description on how to use checkpoint/restart is included in Volume VIII, Job Preparation Manual.

Section 6

OPERATOR LIST AND SYSTEM FLOW

6.1 Consolidated List of Operators

The following list of operators provides a quick reference for the analyst.

a. Run Initializing

FILE

LIMIT

b. Library Action

ADD

REPLACE

DELETE

c. Retrieval Initializing

TITLE

FILE

LIMIT

NO-GO

d. Conditional Statements

KEYWORD

LIMIT

IF

FURTHER

e. Imperative Statements

SORT

SELECT

f. Comment

NOTE

6.2 System Flow

RASP has three sections: the Input Processor section, the Retrieval Proper section, and the Sort section. The flow within each section is discussed briefly in the following paragraphs.

The Input Processor section reads the input stream. It edits each statement and restructures communication records for the rest of the component. It extracts the required information from the file format table, resolves all field name addresses and references, and places those data values converted by conversion subroutines in the communication records. It also applies an algorithm to restructure the conditional logic into a bit pattern which can be more readily interrogated using programmed logic. Finally, it converts the retrieval statements into macros which are assembled and link edited by OS/360 to produce an executable load module. The resulting executable retrieval is stored on the Permanent or Temporary Retrieval Library.

The Retrieval Proper section reads the communication records, calls and executes the code stored on the Permanent or Temporary Retrieval Library. It checks to see if the retrieval, the required subroutines, and the first data set to be retrieved against are available. If either of the first two are not available, an advisory message is issued, and the job is terminated. If the data set is not available, the operator is notified, and appropriate operator action must be taken.

After all initialization has been completed, the data base logical records are read in one at a time. The control field is tested successively against the criteria for the primary limit, if any; the secondary limit, if any; and the references to the fixed set, if any. If the record fails on any of these criteria, all subsequent logical records up to the next fixed set (i.e., the whole logical block) are bypassed. This 'early-fail' feature permits more efficient program operation. The same early-fail technique is also applied for each change in periodic set.

The flow sequence continues until every affected set has been processed against the logic of all retrievals. If, after evaluation, the current logical block qualifies against the conditions of at least one retrieval, the logical block is written on the QDF. The ROAER entries containing sort keys and QDF pointers are generated as required by the Sort and Select statements and processing resumes as before.

The Sort section invokes the OS/360 Sort/Merge program to sort the records in the QRT.

Section 7

RASP SAMPLE RUN

Problem 1

Execute retrievals TSF0312, TSF0301, and TSF0902.

Problem 2

Retrieve all Air Force and Navy records for processing by RIT TFØ311R. Produce additional answer records with the following conditions imposed:

Navy units located in Verdun
Delete all periodic sets except Periodic Set 1
Sort by Activity and Unit Level
Publish under RIT TFØ313R.

Problem 3

Retrieve all Air Force units in Metz, Toulon, and Cannes with tactical air missiles and assigned 2ØØØ or 3ØØØ series plan.

Retain only the fixed set
Preserve file order
Publish under RIT SFØ314R
Produce additional records for processing by a separate RIT SFØ315R that also has a series 5ØXX plan assigned and sort this information in service-unit name sequence; restrict to flagged subsets.

RETRIEVAL AND SORT PROCESSOR (RASP)

SOLUTION

```
//RASP36Ø       JOB      (standard parameters)
//              EXEC     XRASP,ISAM=TEST36Ø,LIB=TEST36Ø,LIBDISP=OLD
//RASP.SYSIN    DD       *
FILE TEST36Ø.
NOTE - THE SOLUTION TO PROBLEM 1 FOLLOWS.
EXECUTE  TSFØ312, TSFØ3Ø1, TSFØ9Ø2.
TITLE    TSFØ315/1, TFØ313R ADD.
    NOTE --   THE FOLLOWING IS THE BASIC LOGIC FOR PROBLEM 2.
    LIMIT IF UIC BT JØØØ1/NØØ3Ø.
    IF SERV EQ NAVY, USAF.
    TFØ311R SORT 'Ø'.
    NOTE -    THE FOLLOWING DEFINES ADDITIONAL LOGIC
              CONDITIONS IMPOSED ON PROBLEM 2.

    FURTHER IF SERV EQ NAVY AND LOC EQ VERDUN.
    TFØ313R SORT ACTIV ## UNLVL.
    SELECT MEMOD.
TITLE    TSFØ316/1, SFØ314R, ADD.
    NOTE -    THE FOLLOWING IS THE SOLUTION FOR PROBLEM 3.
    IF SERV EQ USAF AND LOC EQ METZ,
    TOULON, CANNES AND MEPSD EQ TMS AND
    PLAN BT 2$ AND 3$.
    SFØ314R SORT '  '.
    SELECT UNTYY.
    NOTE -    THE FOLLOWING IMPOSES ADDITIONAL LOGIC
    CONDITIONS ON THE IF STATEMENT.
    FURTHER PLAN EQ 5Ø$.
    SFØ315R SORT SERV #OCMDS#, UNAME.
    SELECT PLEAC/F.
/*
                    'NOVEMBER, 1972'.
```

Figure 1. RASP Sample Run

DISTRIBUTION

CCTC CODES	COPIES
C124 (Reference and Record)	3
C124 (Record Copy) Stock	6
C240	20
C315	1
C341 (Maintenance Contractor)	10
C341 (Stock)	70

EXTERNAL

Director of Administrative Services, Office of the Joint Chiefs of Staff Attn: Chief, Personnel Division, Room 1A724, The Pentagon Washington, D.C. 20301	1
Director for Personnel, J-1, Office of the Joint Chiefs of Staff, Attn: Chief, Data Service Office, Room 1B738C, The Pentagon, Washington, D.C. 20301	1
Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: P & AD, Room 2B870, The Pentagon, Washington, D.C. 20301	1
Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: Deputy Director for Operations (Reconnaisance and Electronic Warfare) Room 2D921, The Pentagon, Washington, D.C. 20301	1
Director for Logistics, J-4, Office of the Joint Chiefs of Staff, Room 2E828, The Pentagon, Washington, D.C. 20301	1
Chief, Studies Analysis and Gaming Agency, Attn: Chief, Force Analysis Branch, Room 1D928A, The Pentagon, Washington, D.C. 20301	1
Automatic Data Processing, Liaison Office National Military Command Center, Room 2D901A, The Pentagon, Washington, D.C. 20301	1

EXTERNAL	COPIES
Automatic Data Processing Division Supreme Headquarters Allied Powers, Europe Attn: SA & P Branch, APO New York 09055	1
Director, Defense Communications Agency, Office Of MEECN System Engineering, Attn: Code 960T, Washington, D.C. 20301	1
Director, Defense Communications Engineering Center, Hybrid Simulation Facility, 1860 Wiehl Avenue, Reston, VA 22070	1
Director, Defense Intelligence Agency Attn: DS - 5C2 Washington, D.C. 20301	5
Commander-in-Chief, Pacific, Attn: J6331, FPO San Francisco, 96610	1
Commander-in-Chief, US Army Europe and Seventh Army ATTN: OPS APO New York 09403	1
Commanding General, US Army Forces Command, Attn: Data Support Division, Building 206, Fort McPherson, GA 30303	1
Commander, Fleet Intelligence Center, Europe, Box 18, Naval Air Station, Jacksonville, Florida 32212	1
Commanding Officer, Naval Air Engineering Center, Ground Support Equipment Department, SE 314, Building 76-1, Philadelphia, PA 19112	1
Commanding Officer, Naval Security Group Command, 3801 Nebraska Avenue, N.W. Attn: GP22, Washington, D.C. 20390	1
Commanding Officer, Navy Ships Parts Control Center, Attn: Code 712, Mechanicsburg, PA 17055	1
Headquarters, US Marine Corps, Attn: System Design and Programming Section (MC-JSMD-7) Washington, D.C. 20380	1

EXTERNAL	COPIES
Commanding Officer, US Army Forces Command Intelligence Center, Attn: AFIC-PD, Fort Bragg, NC 28307	1
Commander, US Army Foreign Science and Technology Center, Attn: AMXSJ-CS, 220 Seventh Street NE, Charlottsville, VA 22212	1
Commanding Officer, US Army Security Agency, Command Data Systems Activity (CDSA) Arlington Hall Station, Arlington, VA 22212	1
Commanding Officer, US Army Security Agency Field Station - Augsburg, Attn: IAEADP, APO New York 09458	1
Commander, Fleet Intelligence Center, Atlantic, Attn: DFS, Norfolk, VA 23511	1
Commander, Fleet Intelligence Center, Pacific, Box 500, Pearl Harbor, HI 96860	1
Air Force Operations Center, Attn: Systems Division (XOOCSC) Washington, D.C. 20301	1
Commander, Armed Forces Air Intelligence Training Center, TTMNIM (360 FFS), Lowry AFB, Co 80230	1
Commander, Air Force Data Services Center, Attn: Director of System Support, Washington, D.C. 20330	1
Commander-in-Chief, US Air Forces in Europe, Attn: ACDI APO New York 09332	1
Commander, USAF Tactical Air Command, Langley AFB, VA 23665	1
Commander, Space and Missile Test Center, Attn: (ROCA) Building 7000, Vandenberg, AFB, CA 93437	1

EXTERNAL	COPIES
Naval Air Systems Command, Naval Air Station, Code 13999, Jacksonville, Florida 32212------	1
Commanding General, US Army Computer Systems Command, Attn: Support Operations Directorate, Fort Belvoir, VA----------	1
Defense Documentation Center, Cameron Station, Alexandria, VA 22314----------	12
TOTAL	159

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: CSM UM 15-78, Volume IV	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): NMCS Information Processing System 360 Formatted File System (NIPS 360 FFS) - Users Manual Vol IV - Retrieval and Sort Processor (RASP)	5. TYPE OF REPORT & PERIOD COVERED / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): DCA 100-77-C-0065 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS: International Business Machines, Corp. Rosslyn, Virginia	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: National Military Command System Support Center The Pentagon, Washington, D.C. 20301	12. REPORT DATE: 1 September 1978 / 13. NUMBER OF PAGES: 142
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.
This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume defines the capability of the Retrieval and Sort Processor (RASP) component of NIPS 360 FFS. It describes the features, functions, restrictions, language qualifications, and expected output results of this component.

This document supersedes CSM UM 15-74, Volume IV.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)